INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Adaptation and Localization of Open Source ERPs - The case of Parties in OFBiz

JORGE MIGUEL ROCHA ALMEIDA
Outubro de 2015

POLITÉCNICO
DO PORTO

Adaptation and Localization of Open Source ERPs

The case of Parties in OFBiz

Jorge Miguel Rocha Almeida

Dissertation to obtain the Master of Science degree
in Computer Science, Specialization in
Architectures, Systems and Networks

Supervisor: Alexandre Bragança

Jury:

President:

[Nome do Presidente, Categoria, Escola]

Vowels:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, October 2015

Resumo

O software tem vindo a tornar-se uma parte importante de qualquer empresa, cobrindo várias áreas funcionais, tais como manufaturação, vendas ou recursos humanos. O facto de uma empresa possuir um software capaz de ligar todas ou a maior parte das suas áreas funcionais e de acomodar as suas regras de negócio permite que estas tenham acesso a dados em tempo real nos quais se podem basear para tomar decisões. Estes tipos de software podem ser categorizados como *Enterprise resource planning* (*ERP*). Tendo em conta que estes tipos de software têm um papel importante dentro de uma empresa, a aquisição dos mesmos é algo que deve ser bem estudado. As grandes empresas normalmente optam pela aquisição de soluções comerciais uma vez que estas tendem a ter mais funcionalidades, maior suporte e certificações. Os *ERPs* comerciais representam, no entanto, um esforço elevado para que a sua compra possa ser feita, o que limita a possibilidade de aquisição dos mesmos por parte de pequenas ou médias empresas. No entanto, tal como acontece com a maior parte dos tipos de software, existem alternativas *open-source*.

Se nos colocássemos na posição de uma pequena empresa, a tentar iniciar o seu negócio em Portugal, que tipo de ERP seria suficiente para os nossos requisitos? Teríamos que optar por comprar uma solução comercial, ou uma solução *open-source* seria suficiente? E se optássemos por desenvolver uma solução à medida? Esta tese irá responder a estas questões focando-se apenas num dos componentes base de qualquer *ERP*, a gestão de entidades. O componente de gestão de entidades é responsável por gerir todas as entidades com as quais a empresa interage abrangindo colaboradores, clientes, fornecedores, etc. A nível de funcionalidades será feita uma comparação entre um *ERP* comercial e um *ERP open-source*.

Como os *ERPs* tendem a ser soluções muito genéricas é comum que estes não implementem todos os requisitos de um negócio em particular, como tal os *ERPs* precisam de ser extensíveis e adaptáveis. Para perceber até que ponto a solução *open-source* é extensível será feita uma análise técnica ao seu código fonte e será feita uma implementação parcial de um gerador de ficheiros de auditoria requerido pela lei Portuguesa, o SAF-T (PT).

Ao estudar e adaptar a solução *open-source* podemos especificar o que teria que ser desenvolvido para podermos criar uma solução à medida de raiz.

Palavras-chave: ERPs, Gestão de Entidades, Apache OFBiz, SAP ERP

Abstract

Computer software has become an important part of any line of business, covering a wide range of functional areas, such as manufacturing, sales or human resources. The fact that a company has a software that is capable of connecting all or most of its functional areas and accommodate its business rules allows companies to have access to real-time data upon which decisions can be made. These types of software fall in the Enterprise Resource Planning (ERP) category. Given that this software plays an important role within a company, the acquisition of an ERP has to be well thought. Big companies usually opt by commercial ERPs as they tend to have more features, support, and certifications. Commercial ERPs are expensive and small companies don't usually have the money to acquire then, also they might not even need half of what they are buying. As it happens with all types of software there are open-source alternatives.

Putting ourselves in the position of a small company, trying to start a new business in Portugal, what kind of ERP would better suit our needs? Would we be better buying a commercial solution, or the open-source one is enough? What if we develop a custom application that implements only our requirements? This thesis will provide answers to this questions by focusing in one of the core modules of any ERP, the Parties module. The parties module is responsible for the management of all entities that interact with a company, ranging from its employees, to its customers, suppliers, etc. We will compare the features provided by a commercial ERP and an open-source ERP.

As ERPs tend to be generic solutions they also tend to lack more specific business requirements, so they need to be extensible. We will provide a technical analysis of the open-source solution to see how we could extend it to support custom business requirements and then we'll extend it to support the generation of an audit file required by the Portuguese government, the SAF-T PT.

By studying the open-source ERP and extending it we can specify what would need to be developed in order to have a functional Party management module compliant with the Portuguese laws.

Keywords: ERPs, Party Management, Apache OFBiz, SAP ERP

Acknowledgements

Firstly, I would like to thank my family and friends, for their support and patience while I was working on this research.

To my supervisor, Alexandre Bragança, for the assistance and availability along the course of this work.

To my work colleagues at Konkconsulting, that have contributed to my professional and personal success. A special thanks to André Bezerra, António Coelho, Egil Silva and Fábio Ramalho for their support along my academic course.

Table of contents

List of Figures

List of Tables

Acronyms

List of Acronyms

A/F	*Accounting and Finance*
AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
CO	*Controlling*
CRM	*Customer Relationship Management*
DBMS	*Database Management System*
DSR	*Design Science Research*
EAM	*Enterprise Asset Management*
ECA	*Event Condition Action*
EECA	*Entity Event Condition Action*
EEOC	*Equal Employment Opportunity Commission*
FI	*Financial Accounting*
HR	*Human Resources*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
M/S	*Marketing and Sales*
MM	*Material Management*
MMS	*Maintenance Management System*
MRP	*Manufacturing Resource Planning*
MVC	*Model-View-Controller*
PM	*Plant Maintenance*

POS	*Point Of Sale*
PP	*Product Planning*
PS	*Project System*
QM	*Quality Management*
SAF-T	*Standard Audit File for Tax*
SCM	*Supply Chain Management*
SD	*Sales & Distribution*
SECA	*Service Event Condition Action*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

1 Introduction

This chapter provides a background for this thesis, explaining some basic concepts and defining the scope of this research. We will also briefly describe the research method that was used, what are the scientific questions that led to the development of this project and what contributions are provided to the field of computer engineering.

1.1 Context

Enterprises, big or small, have the need to constantly optimize themselves, becoming better and quicker at what they do due to the aggressive competition almost any company has. One of the key aspects for a company success is its organization, their business processes and interconnection with the customers and employees. To achieve this, most companies end up acquiring an Enterprise Resource Planning software (ERP) to help improve their workflows. ERPs are often complex and very expensive thus restricting start-ups or small businesses from acquiring them. For these software solutions to provide value to a business they need to cover a lot of functional areas, such as sales, human resources and finance. With everything connected is easy for employees to have a view of their business in real-time, automate some business processes and enforce business rules. Due to the variety of areas that the software covers and the diversity of the law from country to country not every piece of the software can be used world-wide. Also, given that these solutions are often developed without any specific type of business in mind, it's common for companies to pay extra to have specific business rules implemented on the solution that they bought. On the other hand, there are solutions developed for a specific type of business that often cover all of their requirements. The process of choosing an ERP is critical and companies usually opt for paid solutions. Another common practice is to pay for the development of a custom application that covers exactly the needs of

the business in question. There are also open-source solutions on the market that some enterprises are already adopting. This raises the question, how do these free ERPs compare to the commercial ones?

1.2 Research methodology

The development of this thesis was based on the Design Science Research (DSR) methodology. Design Science Research involves a set of activities that lead to the development of a solution to a relevant technology based problem. The research methodology in question is an iterative and incremental process which comprises the following activities:

- Identification of a problem – identifies a problem that is of interest to the field;
- Definition of objectives for a solution – defines the scope of the produced artifacts;
- Design and development – development of the artifacts;
- Demonstration – application of the artifact on a real-world problem;
- Evaluation – evaluate the produced artifacts with well-executed evaluation methods;
- Communication – share the knowledge with the community.

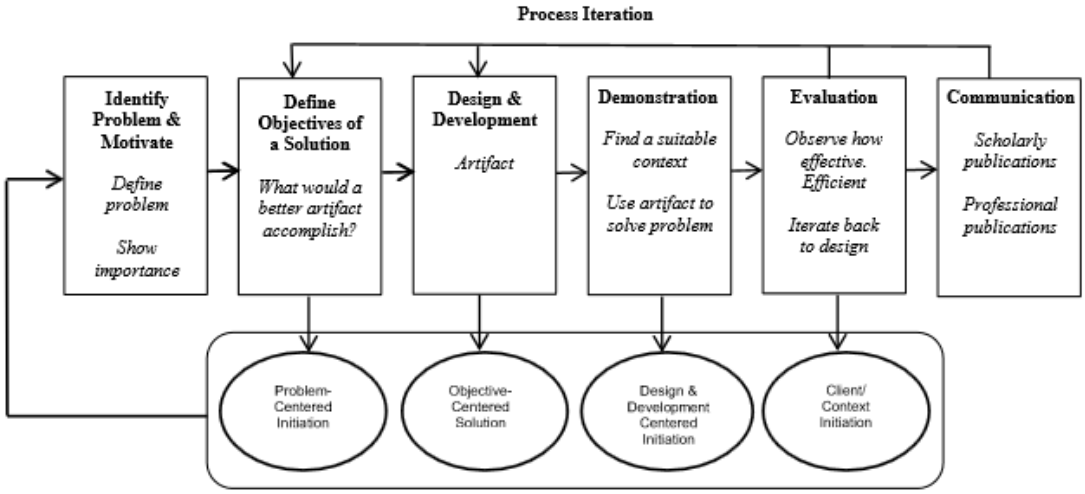


Figure 1 – Design Science Research Methodology Process Model (Vaishnavi & Kuechler, 2013).

According to the Design Science Research in Information systems (Vaishnavi & Kuechler, 2013) the research process can produce one or more of the following general outputs:

- Constructs – Constructs describe the domain of a given problem and/or solution*;*
- Models – "*A model is a set of propositions or statements expressing relationships among constructs."* (Vaishnavi & Kuechler, 2013)*;*
- Methods – A method represents an algorithm and the proceedings of a solution to a given problem or a set of problems;
- Instantiations – *"Instantiations operationalize constructs, models and methods. They are the realization of the artifact in an environment."* (Vaishnavi & Kuechler, 2013).

When it comes to the evaluation of the produced artifacts the Design Alternatives for the Evaluation of Design Science Research Artifacts (Cleven, Gubler, & Hüner, 2009) paper suggests the following methods:

- Action research;
- Case study;
- Field experiment;
- Formal proof;
- Controlled experiment;
- Prototype;
- Survey;

Variable	Value				
Approach	Qualitative		Quantitative		
Artifact Focus	Technical		Organizational		Strategic
Artifact Type	Construct	Model	Method	Instantiation	Theory
Epistemology	Positivism		Interpretivism		
Function	Knowledge function	Control function	Development function	Legitimization function	
Method	Action research	Case study	Field experiment	Formal proofs	
	Controlled experiment	Prototype	Survey		
Object	Artifact		Artifact construction		
Ontology	Realism		Nominalism		
Perspective	Economic	Deployment	Engineering	Epistemological	
Position	Externally		Internally		
Reference Point	Artifact against research gap	Artifact against real world	Research gap against real world		
Time	Ex ante		Ex post		

Figure 2 – Variables and values for the evaluation of DSR artifacts (Cleven, Gubler, & Hüner, 2009).

The artifacts developed during the research must provide a solution to the relevant technology-based problems and the knowledge must be new and true. We will produce one instantiation, the guide to effectively choose between adapting an open-source ERP or creating a fit-to-purpose application. This instantiation will be tested by developing a prototype, a partial implementation of a common business requirement for ERPs that the chosen open-source software does not implement.

1.3 Research questions

For a small or medium company, a commercial ERP has a high cost and may have many unnecessary features. Due to the weak advertisement and lack of certifications of the free solutions companies usually end up with a custom application or a small ERP developed for a specific type of business. This leads to the main question:

RQ1 - *What would be the effort, if any, to adapt an open-source ERP to local law compared to developing a fit-to-purpose application from scratch?*

The following question arises from the main question:

RQ2 - *How does an open-source ERP compare to a commercial one in terms of features?*

1.3.1 Literature questions

In order to provide answers to the questions presented above there are more questions that arise and that can be answered with information taken from the literature available. The first question is:

LQ1 - *What is an ERP and what features should it provide?*

Another relevant question that arises from the first question is:

LQ2 - *How does the data model of an open-source ERP compare to the one of a commercial ERP?*

Since the features available vary from ERP to ERP we will choose one commercial ERP and one open-source ERP for analysis and comparison. For the commercial ERP we chose SAP ERP and for the open-source we chose Apache OFBiz.

1.4 Scope

This thesis aims to act as a guide for anyone that is considering the acquisition of a custom ERP solution and is unsure if adapting an existing open-source solution could be better than developing a fit-to-purpose application from scratch. The concepts behind ERPs will be explained and a comparison between SAP and Apache OFBiz will help clarify what features both have. This thesis will only focus on the Parties module of each ERP as it can be considered the base for most ERPs.

To be able understand if the Apache OFBiz is flexible enough to tackle new requirements we will describe Apache OFBiz from a technical view, explaining the multiple application components and how they work together. Since Apache OFBiz is lacking some of the requirements of the Portuguese law, a prototype implementation of the SAF-T audit file will be developed.

1.5 Contributions

This thesis provides two distinct contributions, first it provides a guide to small or medium companies that are in the process of choosing an ERP solution and want to know if building a custom application is better than learning and adapting an open-source solution. In this case the analysis will be done only for the Parties module of the ERP and in the context of the current Portuguese law.

The second contribution is the development of a technical document that intends to help developers better understand the architecture and the technologies used in the Apache OFBiz ERP software. This contribution is justified by the fact that there is a small amount of documentation available and most of it is either outdated, separated or incomplete.

Both of the contributions stated above will be evaluated by the development of a prototype implementation of the SAF-T PT audit file.

1.6 Document structure

This document contains six main chapters which are organized as follows:

- Chapter 1 presents the motivation behind the research that will be presented in the next chapters as well as the main goals and contributions;
- Chapter 2 is dedicated to better understand what is an ERP, what functional areas they cover, what are the responsibilities of each functional area and how they're connected with each other. After understanding the functional areas, we will describe a generic data model for the Parties management module, which is similar to the data model used by Apache OFBiz that we will present later. Lastly, we will take a brief overview of the SAP ERP, a commercial ERP. The overview will describe the core SAP modules and then we will investigate how SAP manages the parties of a given company by taking a look at some of the application screens and the underlying data model;
- Chapter 3 describes the major application modules of Apache OFBiz and then presents a technical overview of the application. This overview is important to better understand the architecture of OFBiz and how it can be extended;
- Chapter 4 presents the SAF-T audit file (Portuguese version), what information compose the file and what information is going to be exported. Then, this chapter will present the architecture and technical details of the partial implementation of the SAF--T (PT) audit file exporter that was developed for OFBiz;
- Chapter 5 takes into account what was developed in chapter 4 as well as the base capabilities of OFBiz and describes the tasks that would need to be executed in order to develop a party management solution from scratch.
- Chapter 6 presents the conclusions that can be taken from this research. It also presents the future work to further improve the presented artifacts and contributions.

1.7 Summary

In this chapter we gave an introduction to the concept of Enterprise Resource Planning software and to the problem that led to the development of this thesis. We identified the literature questions (LQ1-2) and research questions (RQ1-2) that will be answered, what contributions this thesis will provide to the field of computer engineering and how the contributions will be evaluated.

2 ERPs

2.1 What is an ERP?

ERP is an acronym for Enterprise Resource Planning. According to *Concepts in Enterprise Resource Planning* (Ellen Monk, 2012) ERP systems "*are core software programs used by companies to integrate and coordinate information in every area of the business.*" ERPs can also be considered as a type or class of software. The main goal of any ERP is to integrate information from all areas of business, making it available to the right person at the right time and providing an efficient way to manage the business processes.

A business process is a group of activities that are performed in some order to create a specific output that is of value to the customer. A business process focuses on the order of those activities, how they are conducted, and clearly defines their inputs and outputs. A business process may use information from more than one functional area.

A functional area is a grouping of activities, processes and individuals based on their goals or activities. Most companies have four main functional areas: Marketing and Sales (M/S), Supply Chain Management (SCM), Accounting and Finance (A/F), and Human Resources (HR). Each area is composed of a variety of narrower business functions, which are activities specific to that functional area of operation. Examples of functional areas of operation and business functions are shown on the table below:

Functional area of operation	Marketing and Sales	Supply Chain Management	Accounting and Finance	Human Resources
Business function	Marketing a product	Purchasing goods and raw materials	Financial accounting of payments from customers and suppliers	Recruiting and hiring
	Taking sales orders	Receiving goods and raw materials	Cost allocation and control	Training
	Customer Support	Transportation and logistics	Planning and budgeting	Payroll
	Customer relationship management	Scheduling production runs	Cash-flow management	Benefits
	Sales forecasting	Manufacturing goods		Government compliance
	Advertising	Plant maintenance		

Table 1 – Examples of functional areas of operation and their business functions (Ellen Monk, 2012).

Integration between these functional areas improves the communication and can help in the optimization of business processes. A functional area usually depends on data from another functional area, for example what was sold by the Marketing and Sales is linked to what was produced by the Supply Chain Management. The business process for the sales of a smartphone is illustrated below:

Input	Functional Area responsible for input	Process	Output
Request to purchase smartphone	Marketing and Sales	Sales order	Order is generated
Financial Help to purchase	Accounting and Finance	Arranging financing in-house	Customer finances through the smartphone company
Fulfillment of order	Supply Chain Management	Shipping and Delivery	Customer receives smartphone
Technical support	Marketing and Sales	24-hour help line available	Customer's technical query is resolved

Table 2 – Sample business processes related to the sale of a smartphone (Ellen Monk, 2012).

A typical ERP software contains various modules, each module being focused on a specific functional area.

Sometimes the business requirements of a given company are so unique that the ERP system does not have any built-in functionality to handle the case. The term “Functional Gap” is used to describe those unique business requirements. These functional gaps can be missing relevant information for an entity, a custom business process, or simply a report. Because of this an ERP system must be customizable/extensible.

In most cases, the ERP system must be capable of communicating with third-party software being used on the same company or in an external company, e.g. the supplier’s software and the customers front-end.

2.2 Functional Areas

2.2.1 Marketing and Sales

The marketing and sales department is responsible for increasing the company revenue, growth and profitability. This can be achieved by creating advertisements, publications or events to show new or existing products. This functional area also analyses the sales of existing products to identify new product gaps that may attract new customers. For the Marketing and Sales department it is important to keep record of customers so flyers and newsletters can be sent. Also when it comes to developing new products some of the data from the Supply Chain management could be analyzed in order to determine the profitability more accurately. The information from the Accounting and Finance department is also relevant at the time of an order or sale to check the customer’s financial history with the company, e.g. if the customer has credit or owes money.

The following figure shows the data needed by Marking and Sales from other functional areas to effectively conduct the activities for which it is responsible:

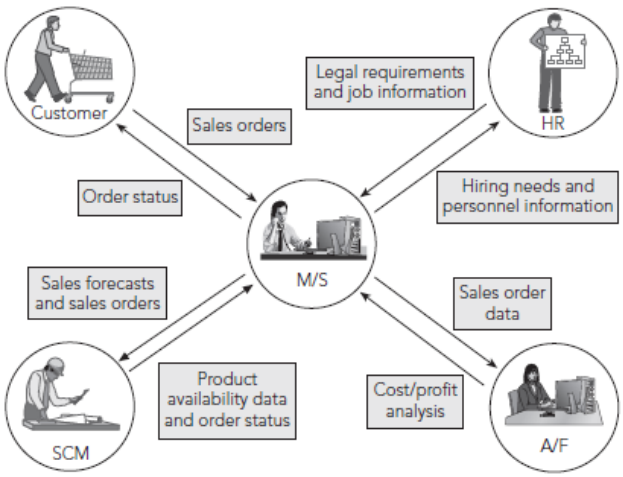


Figure 3 – Integration between the marketing and sales functional area and other functional areas (Ellen Monk, 2012).

2.2.2 Supply Chain Management

Supply Chain Management plays an important role in any business and its optimization is essential for the efficiency of any company. Inefficiency in this department might compromise the satisfaction of customers as well as the company success. SCM is responsible for making sure that all the right products are available at the right location and/or the right resources are available when they are needed. Essentially it is the job of the Supply Chain department to develop production plans and guarantee that the correct raw products are ordered and received at the correct facility as well as check that the correct products are manufactured and shipped to the customers. The Supply Chain Management must have access to the Sales information to determine what resources are needed and where.

The following figure shows the data needed by Supply Chain Management from other functional areas to effectively conduct the activities for which it is responsible:

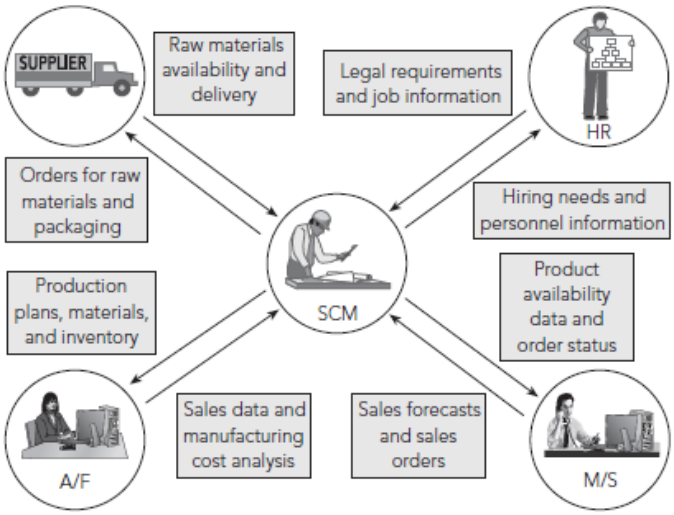


Figure 4 – Integration between the supply chain management functional area and other functional areas (Ellen Monk, 2012).

2.2.3 Accounting and Finance

The Accounting and Finance department is perhaps one of the most interconnected departments within a company. It is responsible for the payroll of every employee, taking into account the salaries earned by every employee, every pay period, etc. The accounting and finance also has the responsibility to verify all cash collections, either from sales or from other sources, all payments made to employees or suppliers and keeping track of all the costs related to the company assets (offices, equipment, etc.). Data from Accounting and Finance can be used by the Marketing and Sales when it comes to forecasting sales for a given period of time as well as by the Supply Chain Management when it comes to developing production plans.

The following figure shows the data needed by Accounting and Finance from other functional areas to effectively conduct the activities for which it is responsible:

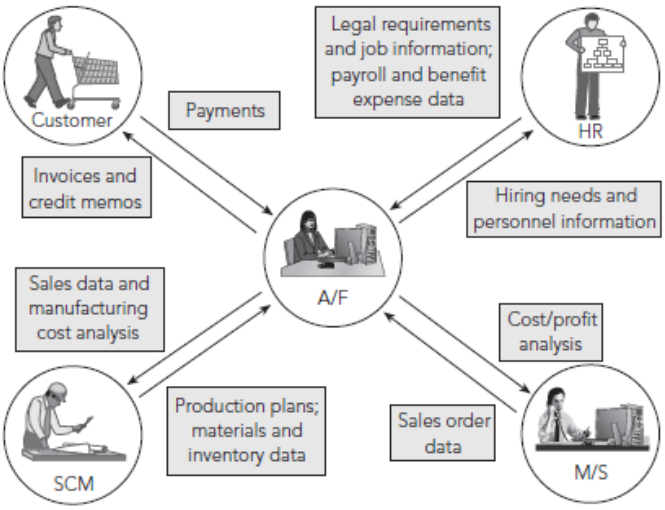


Figure 5 – Integration between the accounting and finance functional area and other functional areas (Ellen Monk, 2012).

2.2.4 Human Resources

The Human Resources department is responsible for managing the company employees as well as recruiting new employees, planning their career and appraising their performance. The HR department also determines the wages and salaries for the different positions within the company. The HR department must sometimes look into the data from the Sales and Supply Chain management when it comes to deciding if a new employee should be hired or not.

The following figure shows the data needed by Human Resources from other functional areas to effectively conduct the activities for which it is responsible:

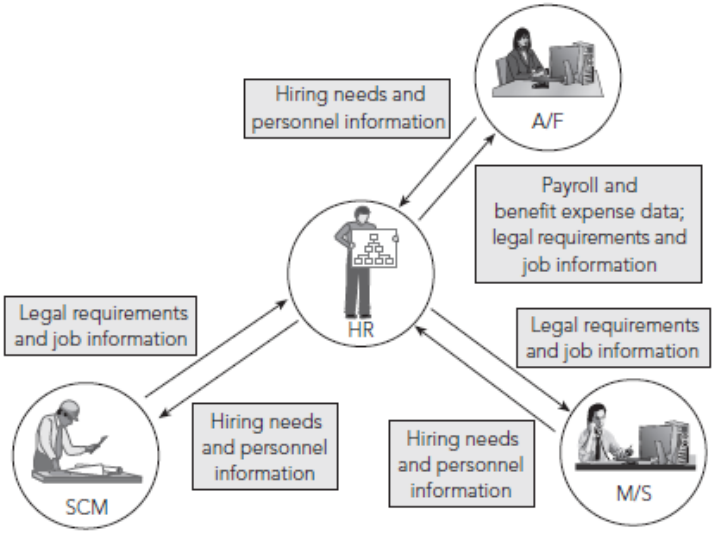


Figure 6 – Integration between the human resources functional area and other functional areas (Ellen Monk, 2012).

2.3 Generic Party Data Model

The Party module covers a part of the responsibilities of the Human Resources functional area but it is not restricted to that. Not only does it manage employees, it also manages customers, suppliers and all other entities. The party module is responsible only for managing the information tied to all parties that the company interacts with. This information ranges from postal address information, to security permissions within the application itself. In this chapter we will take a look at a generic data model that can be found in *The Data Model Resource Book* (Silverston, 2001). This generic model is important as it is the base for the data model used by OFBiz. The figure below contains the conventions for the data models displayed in the current chapter.

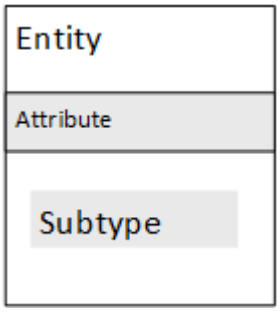


Figure 7 – Data model conventions

2.3.1 Party Entity

Party is an entity that can describe organizations or people. It is used to store the information that both entities have in common, such as email address or phone number. Organizations and people can also have common roles within a company, they can both be sellers or customers. If these two entities were stored separately, the data model would require two relationships for any transaction that could involve a person or an organization, e.g. a sales order, and these relationships would need to be mutually exclusive (e.g. a sales order cannot be placed by a person and an organization at the same time).

Parties are also classified using the Party Classification entity. The Party Classification entity holds only the information shared between the Organization Classification and Person Classification entities. There are subtypes of classifications for both types of party classifications. For the Organization Classification the subtypes can be, for example:

- Industry Classification
  Possible values can be: telecommunications, government institute, manufacturer.
- Size Classification
  Possible values can be: small, medium, large.

The subtypes that can be used to categorize people can be, for example:

- EEOC (Equal Employment Opportunity Commission) Classification

Possible values can be: African American, Native American, Asian or Pacific Islander, Hispanic, and white non-Hispanic.

- Income Classification
  Possible values: less than €20,000, €20,001 to €50,000, €50,001 to €250,000, and over €250,000.

These subtypes can be used for market segmentation, to target marketing efforts and to determine if there are any special pricings, etc.

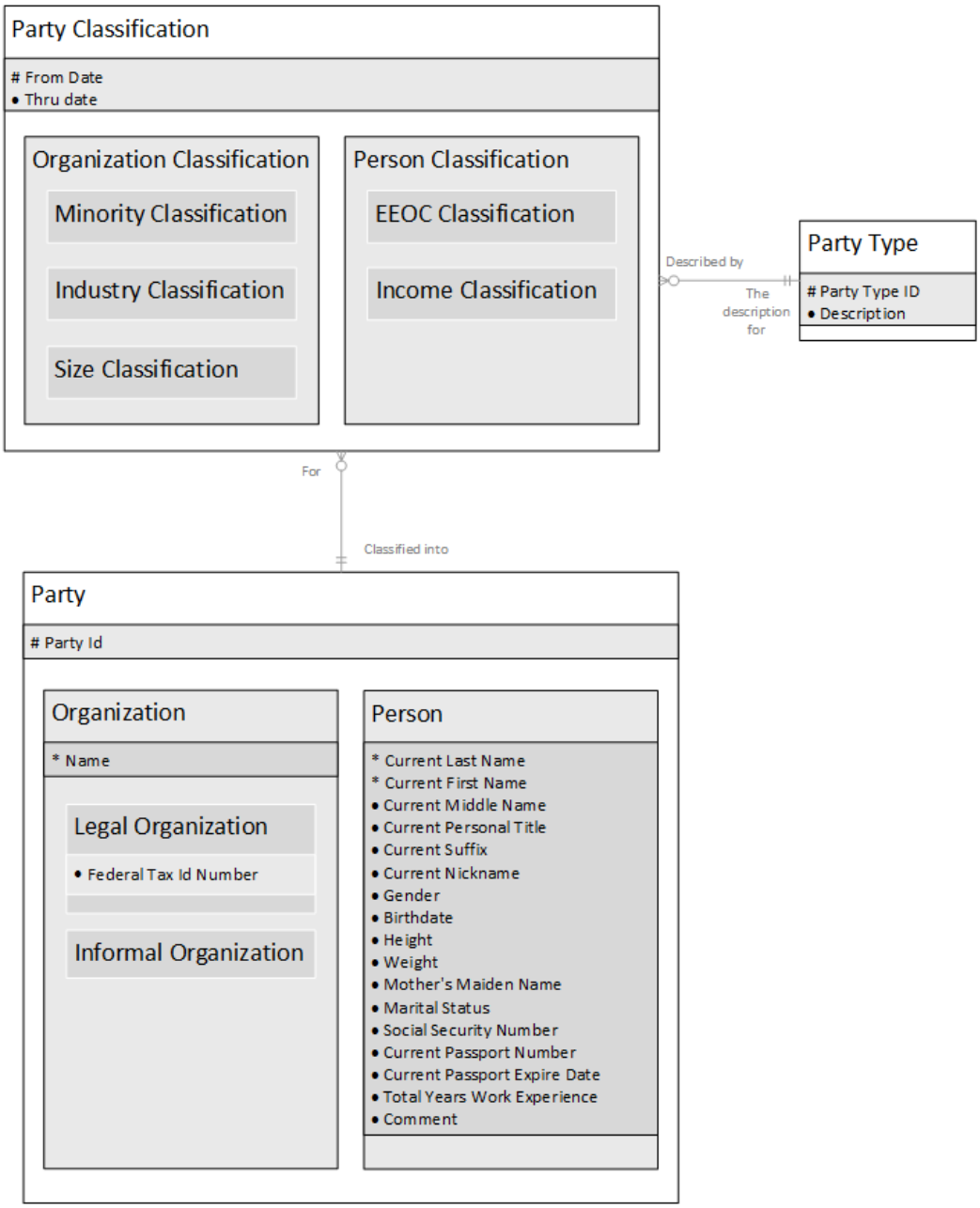


Figure 8 – Party data model (Silverston, 2001).

2.3.2 Party Roles

As said before people and organizations can have common roles within a company, such as customer, supplier, employee, or internal organization. They can also have one or more of these roles at any point in time or have many roles over time. The Party entity is used to define the information that does not change regardless of the party role. The Party type classifies the party into categories. The Party role defines what roles the party plays in the company environment. There's information that is only applicable for a given role, e.g. an employment contract can only be related to an employee and thus the relationship between the table that holds the employment contracts and parties has to check that the party has the employee role.

The Party roles can be divided into distinct groups:

- Person Roles
  A person role can be for example: employee, contractor, family member etc.
- Organization Roles
  Organization roles are used to describe the roles of internal or external organizations. These roles can include for example the following values: Competitor, Partner, Regulatory Agency, Subsidiary, Department Division and other Organization Units or Internal Organizations.
- Common Roles
  Common roles can be seen as either a person role or organization role, for example: the Customer role.

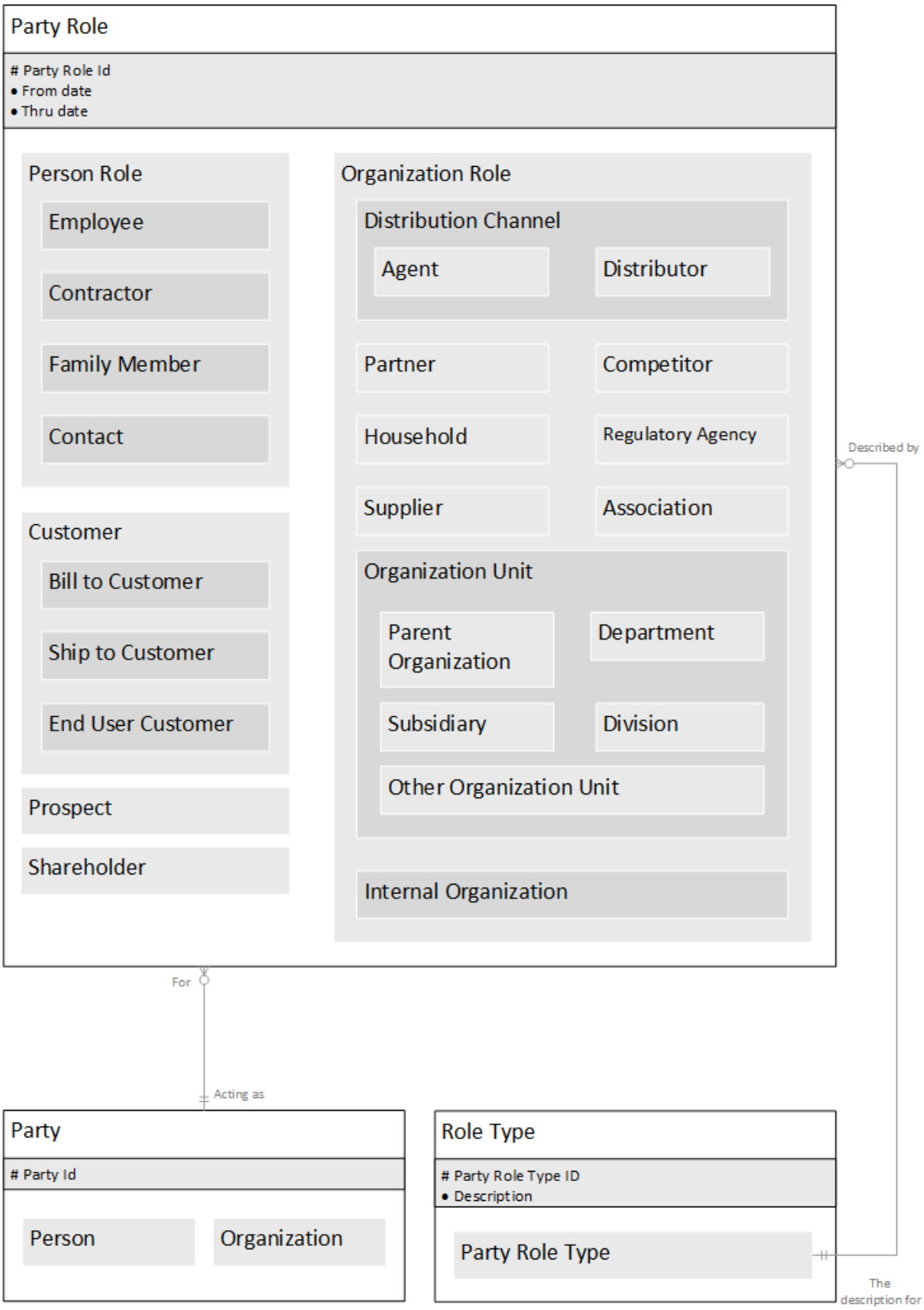


Figure 9 – Party roles data model (Silverston, 2001).

2.3.3 Party Relationship

Parties can have multiple relationships with other parties within the company environment, for example we have the relationship between an employee and the company as an Employment relationship and we can have a relationship between an external party and the company as customer. The party relationship entity allows parties to be related to other parties, and clearly states their respective roles. Examples of party relationship values can be:

- Employment
  The employment relationship provides a way to identify people who are employees of an internal organization
- Customer
  This party relationship allows a customer to be related with several internal organizations as a customer
- Organization rollup
  The organization rollup shows that an organization unit can be within another organization unit.

The party relationship entity also holds additional information of the relationship. For a relationship we can specify a priority, the status and multiple communication events. The priority type states the importance of the relationship to the company, example values can be: “high importance”, “medium importance” and “low importance”. The relationship status is used to specify the current state of the relationship. Example values can be: “active” or “inactive”. Each communication event related with a party relationship is used to record any type of contact between the two parties, for example, an email, a phone call or a meeting.

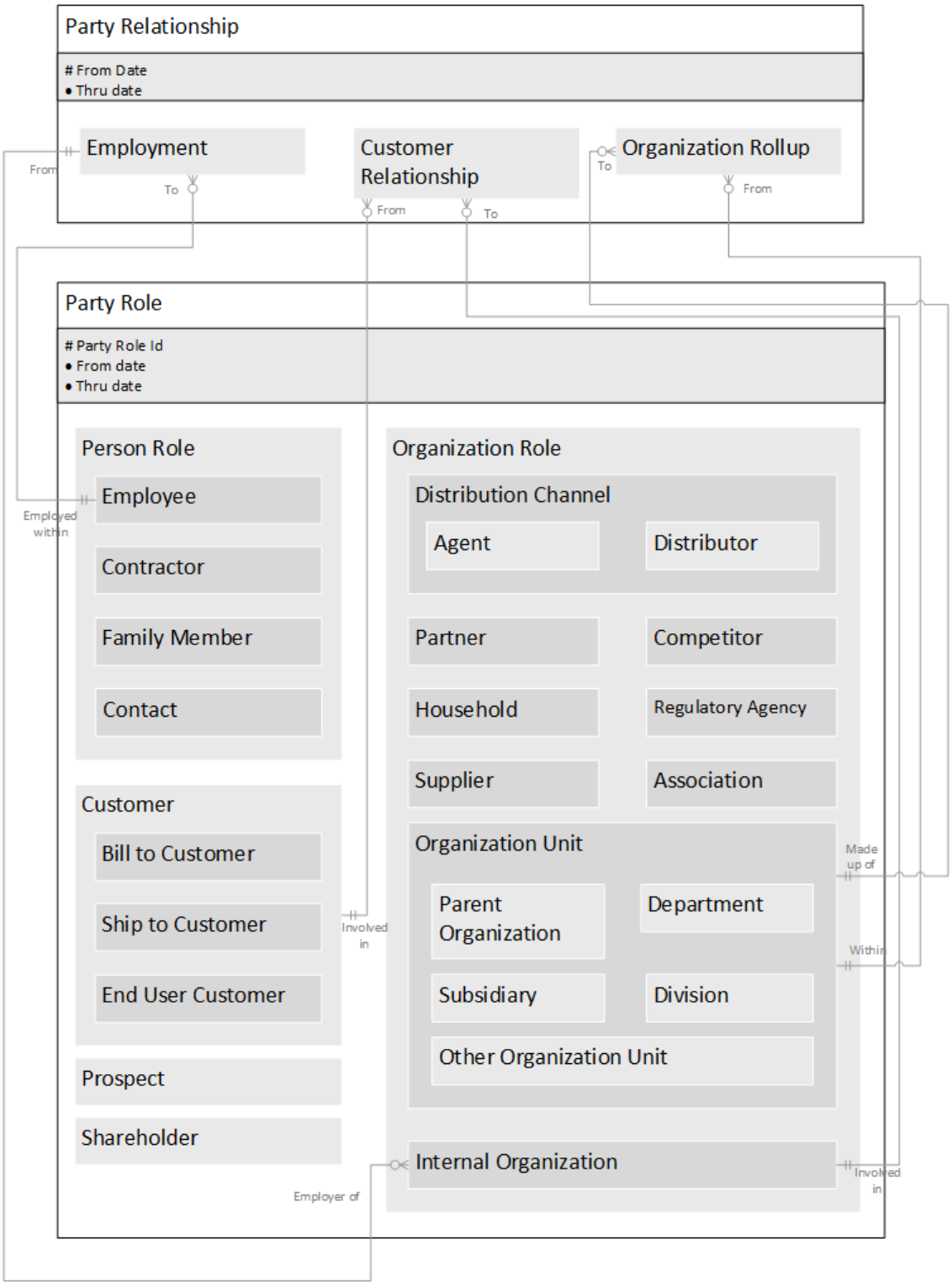


Figure 10 – Specific party relationship data model (Silverston, 2001).

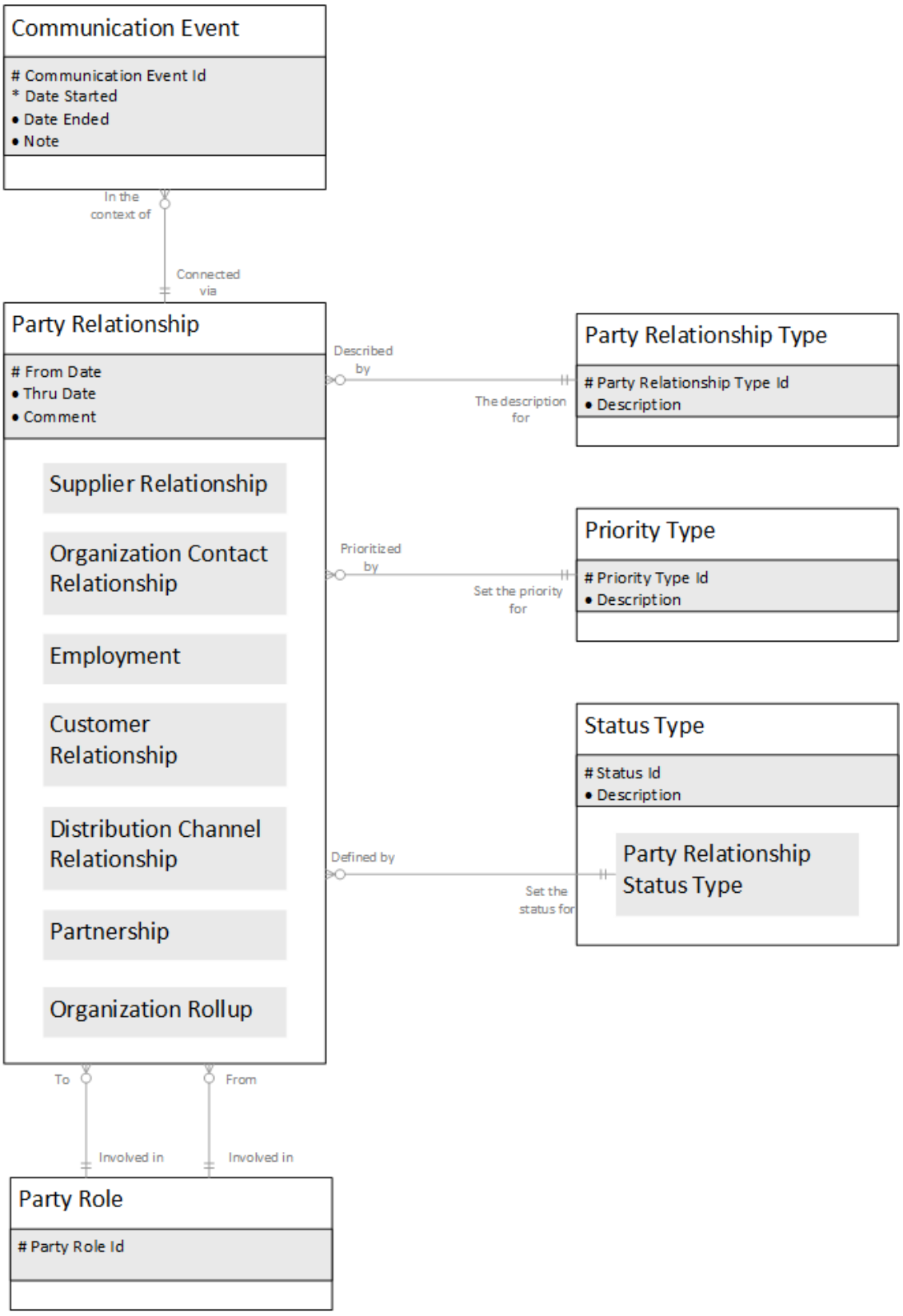


Figure 11 – Party relationship information (Silverston, 2001).

2.3.4 Party Contact Information

As said before, parties can be related to each other and communication between them is important. For that to happen, the contact information of a given party needs to be stored, for example: the phone number or email address.

2.3.5 Postal Address Information

The Postal address entity maintains the addresses for all enterprise parties in a central place. The Postal address specifies which postal addresses are related to a party as well as the Geographic Boundaries encompassing the area such as country, city, state, etc.

The party postal address entity needs to be capable of storing multiple postal addresses as an organization can be available at more than one location, for example a supplier can have more than one warehouse. The opposite can also happen, multiple organizations can have the same address if they share the same office.

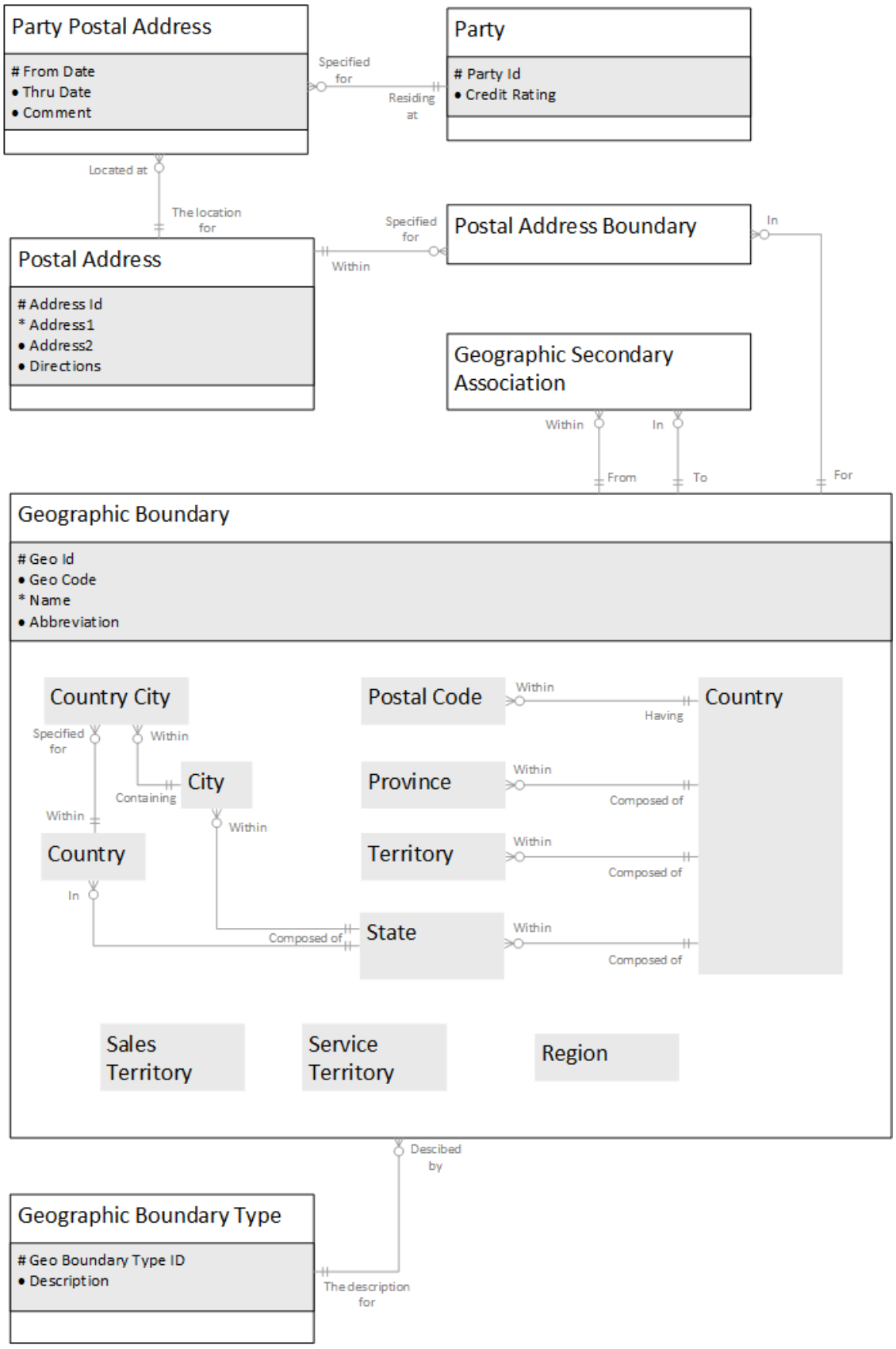


Figure 12 – Postal Address data model (Silverston, 2001).

2.3.6 Party Contact Mechanism

The contact mechanism entity contains details regarding how an entity can be contacted, e.g. by telephone or email address. To connect the contact mechanism and parties there is an intersecting entity, the party contact mechanism. The contact mechanism is also related to a contact mechanism type that can hold valuable information for a given type. Examples of contact mechanism types can be: phone, fax, mobile phone, email address, etc. The party contact mechanism model can also be expanded to store the address, if needed. A contact mechanism can also be of a specific location, for example the telephone number for a manufacturing plant. These physical locations require a new entity to describe them as they are not addresses nor parties (event though they are associated with the postal address and party entity). To accommodate the existence of facility contacts the contact mechanism entity contains a Facility field that stores information or relationships related with these locations.

Each facility can hold one or more Parties, the role that each of the parties have in the facility are stored in the Facility Role field. Also each facility can be composed of other facilities, for example, an office building can have more than one room. These types of facilities are distinguished by the Facility Type entity.

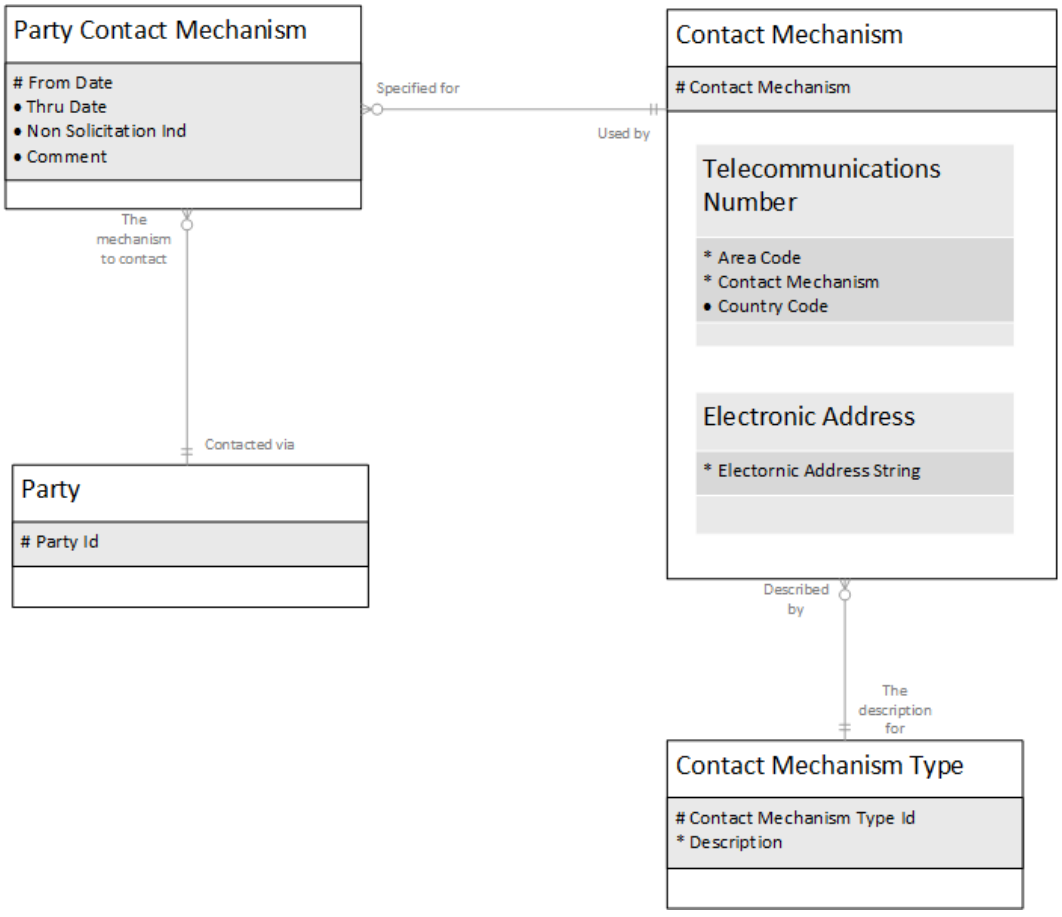


Figure 13 – Party contact mechanism data model (Silverston, 2001).

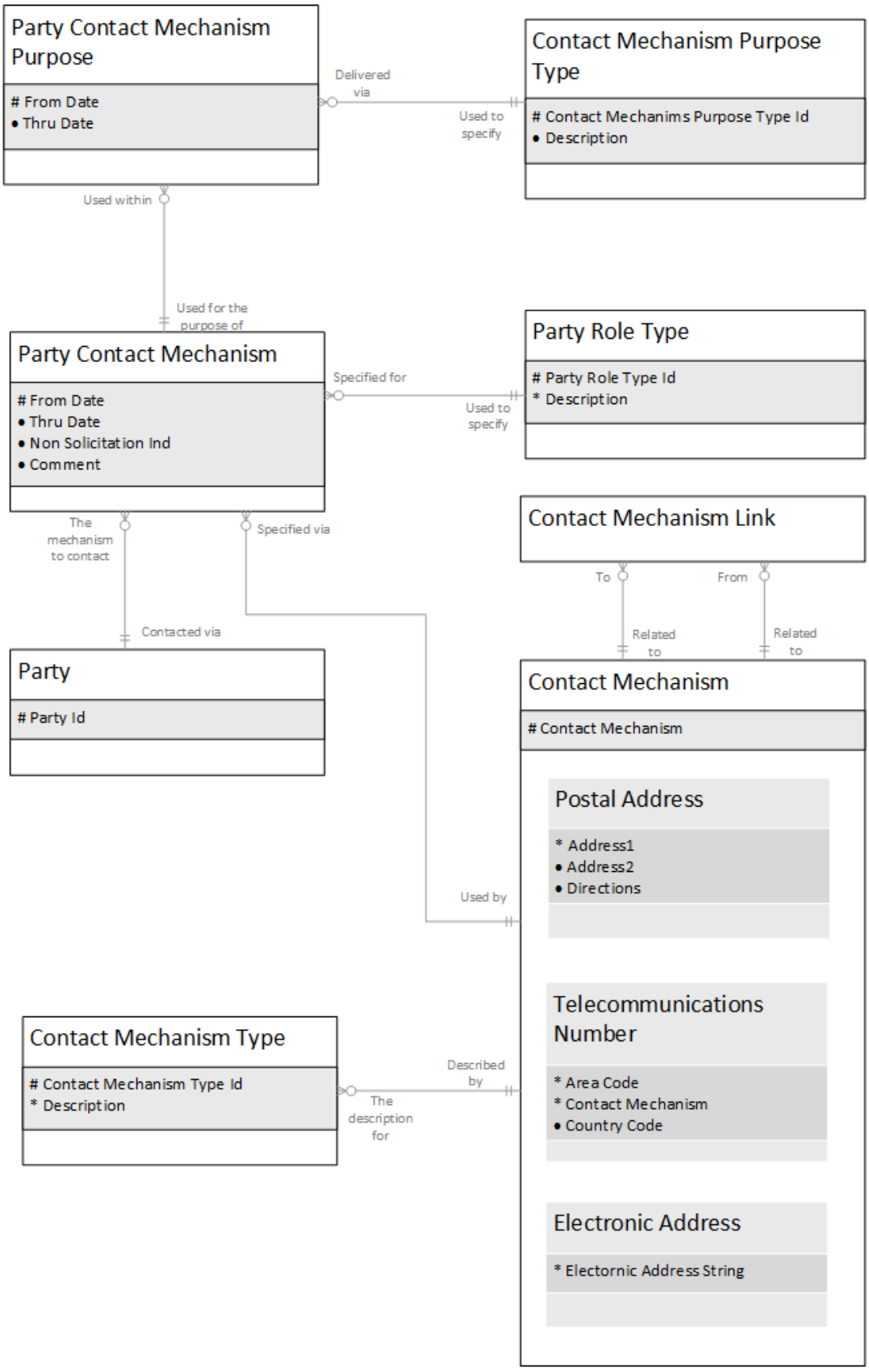


Figure 14 – Party contact mechanism expanded data model (Silverston, 2001).

2.3.7 Contact Mechanism Purpose

A party can have one or more contact mechanism and each one may serve a different purpose. To distinguish what purpose a contact mechanism can have the contact mechanism entity needs to be associated with one or more contact mechanism purposes. These purposes can be for example: Home phone, Work phone, billing address, home address, etc.

2.3.8 Party Communication Event

In a lot of companies, it is important to maintain a history of what communications where made with a Party, what was its purpose, who was responsible for them and when did they happen. A Communication Event provides a mechanism to store communications that occurred or will occur between a set of parties. This communication event can happen in the context of a party relationship or between many parties (e.g. a seminar). The communication event role is used to specify the role each party had in the communication event (e.g. facilitator, participant, etc.). A communication event requires one and only one Communication Mechanism Type and it may have one or more Communication Event Purposes. Since the contact mechanism types can have different values, there is a need to specify what roles can be assigned to a party in a communication event for each contact mechanism type, for example: caller and receiver do not make sense for a face-to-face meeting. The Valid Contact Mechanism Role entity is used to specify what Communication Event Role Types are valid for a given Contact Mechanism Type. A status of the communication is also stored by the Communication Event Status Type entity. The status can be, for example: “scheduled”, “in progress” or “completed”.

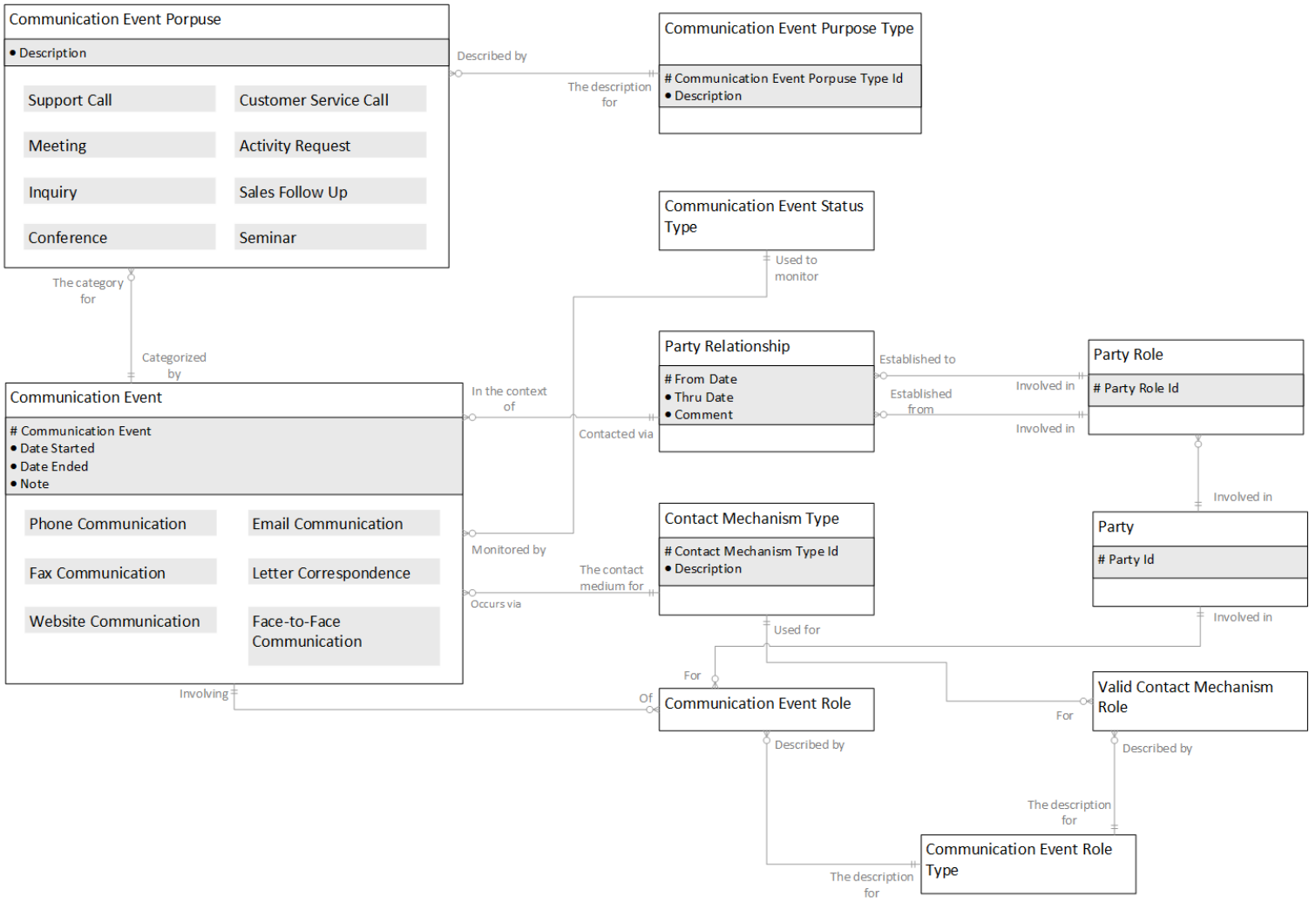


Figure 15 – Party communication event data model (Silverston, 2001).

2.3.9 Communication event follow-up

It is common for a communication event to require an action from one of the involved parties, e.g. a support call might require an action to be taken from the employee such as sending the latest patch, or sending a team to help with the issue. For the data model to support these cases a new entity is required, the Case entity. The case entity acts as a group of communication events related with each other. Each case may have several Case Roles used to distinguish the roles of the parties involved in the case (quality of service, customer, etc.). Each communication event can also have a related Work Effort, which defines the tasks that need to be done as a result of the communication event. A Work effort can be associated with multiple Communication events.

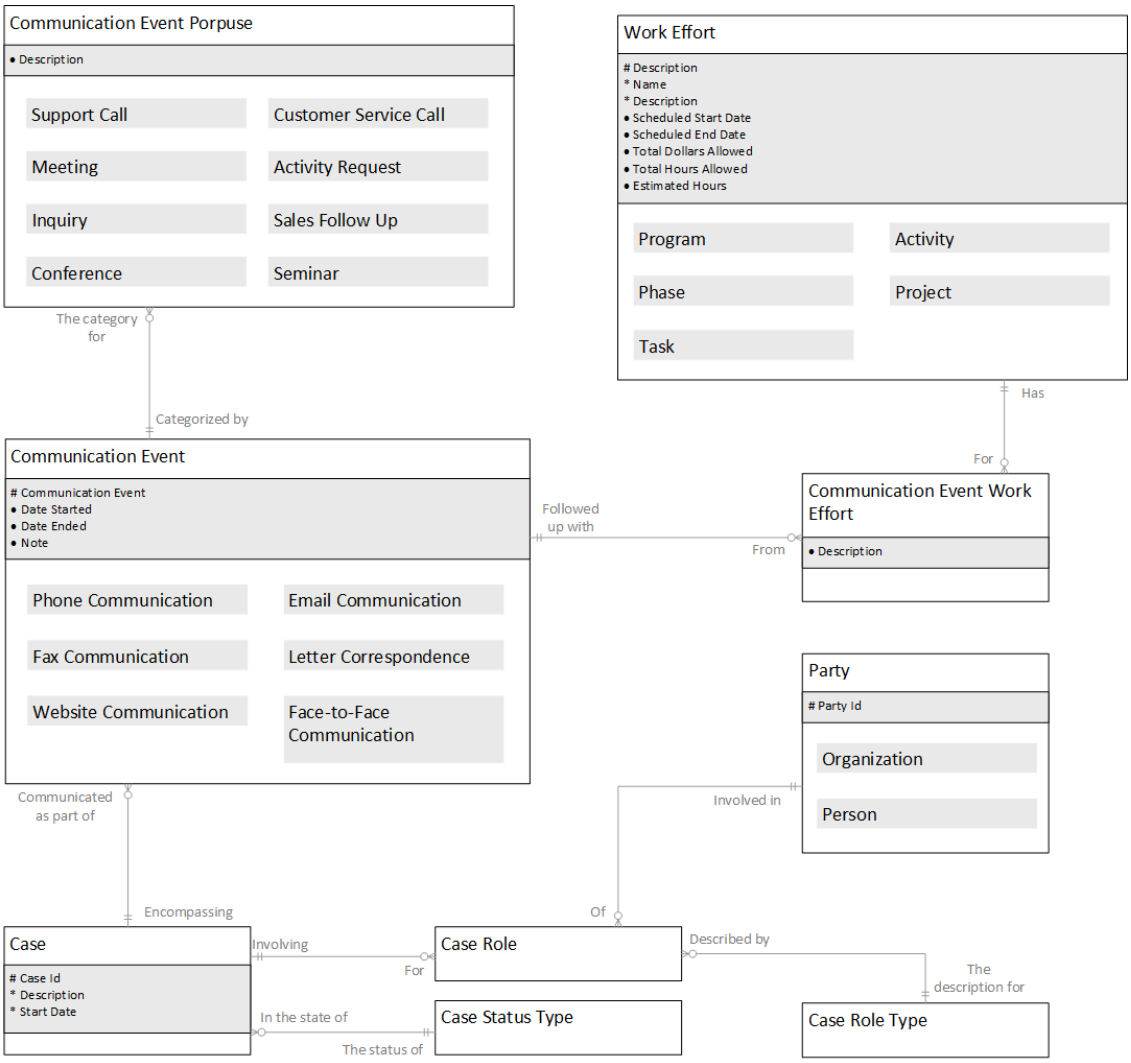


Figure 16 – Communication event follow-up data model (Silverston, 2001).

2.4 SAP ERP

In this chapter we will take a look at the SAP ERP modules and how it relates with the functional areas seen above. We will also briefly describe the data model used by SAP to manage parties. SAP ERP was chosen because it was one of the commercial ERPs with higher market share by the end of 2013, according to Gartner[1] (Columbus, 2014).

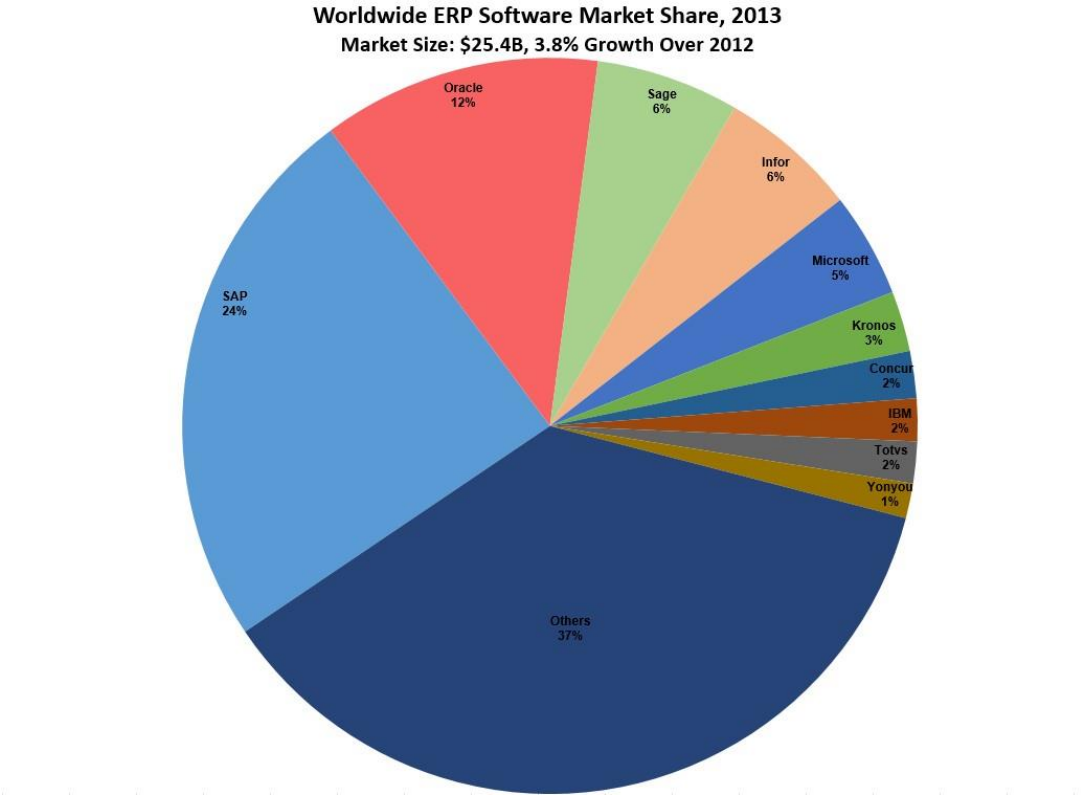


Figure 17 - Worldwide ERP Software Market Share, 2013 (Columbus, 2014).

The fact that SAP has been around much earlier[2] than Apache OFBiz and the fact that it has a data model that differs from the Generic Party Data Model, as we will see later, makes it an interesting candidate to be studied.

2.4.1 Main SAP Modules

SAP ERP is a commercial enterprise resource planning software developed by SAP SE[3]. The ERP has 12 main modules which are composed of multiple sub-modules. In this chapter the main components will be described.

[1] https://www.gartner.com/doc/2477517/market-share-analysis-erp-software
[2] http://www.sap.com/corporate-en/about/our-company/history/1972-1981.html
[3] http://go.sap.com/index.html

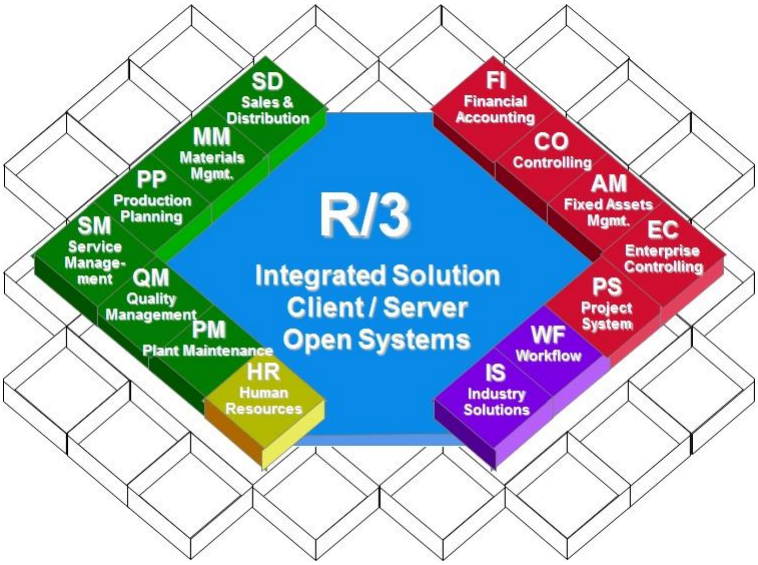


Figure 18 – Main SAP Modules. (Beginners-SAP, 2015)

- Financial Accounting (FI)
  The FI module is an important core module developed to meet all the accounting and financial requirements of an organization.

- Controlling (CO)
  The CO module is intended to be used by Management for the purpose of planning, reporting and decision making. It represents a high level of information regarding the company's flow of cost and revenue.

- Sales & Distribution (SD)
  The SD module is part of the logistics module that support customers. This module covers quotations, sales orders and all the business flow towards billing the customer. This module is highly integrated with the Material Management and Product Planning modules.

- Material Management (MM)
  This module is responsible for the Procurement Handling and Inventory Management. It is integrated with the Sales & Distribution, Product Planning and Quality Management Modules.

- Product Planning (PP)
  The purpose of the PP module is to plan and manage the manufacturing process with the objective of producing effectively and efficiently the customer's products.

- Quality Management (QM)
  The QM module offers tools to ensure that all the activities executed during the life-cycle of a product are effective and efficient with respect to the system and its performance.

- Plant Maintenance (PM)
  The PM module was developed to cover all the supporting activities of a given Plant such as availability of equipment, safety control or inventory management.

- Human Resources (HR)
  The HR module is responsible for supporting all the activities done by the human resources department of an organization, such as personnel administration, organizational management, recruitment, etc.

- Project System (PS)
  The PS module supports the planning, control and monitoring of projects within a company.

In the SAP ERP the information is separated into several Infotypes, each of these infotypes is directly related to a database table. An infotype is basically an aggregator of information. In order to facilitate data maintenance, the infotypes can be combined into infogroups. Each infotype can be identified by its name or by a four digit number.

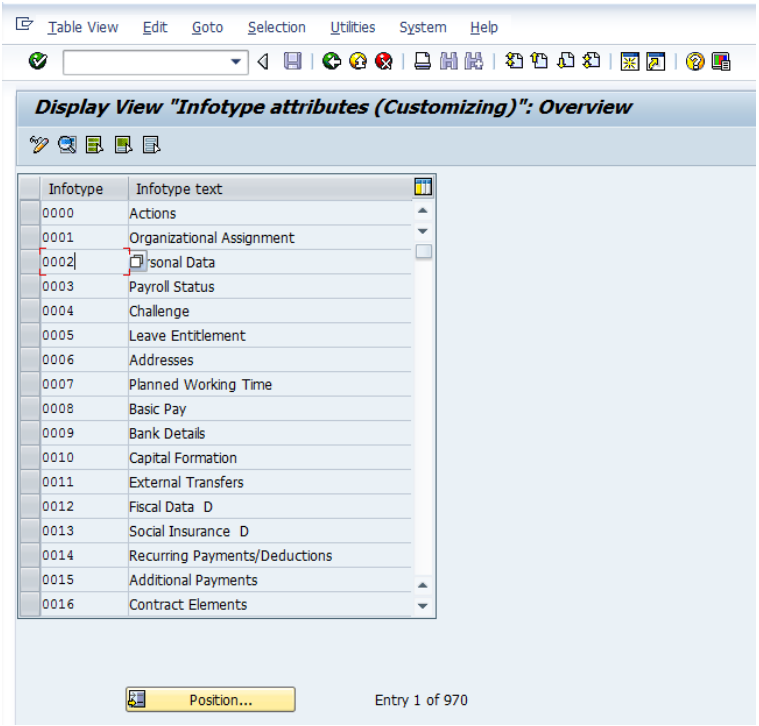


Figure 19 – SAP Infotypes.

2.4.2 Main Infotypes of the Human Resources module

In the HR module an infotype is the combination of professional data which belongs together, e.g., addresses, bank details, additional payment, etc. An infotype can contain field validations such as obligatory fields.

As said before, an infotype can be identified by a four digit number. The following list shows the infotype number ranges:

- 0000-0999 – personnel administration (or recruitment);
- 1000-1999 – personnel planning and development;
- 2000-2999 – time management;
- 3000-3999 – logistics integration;
- 4000-4999 – exclusively recruitment;
- 9000-9999 – customer-specific infotypes.

2.4.2.1 Subtypes

Subtypes partition an infotype into screens of similar content. An example of this is Infotype 0009 ("Bank Details") with the subtype "Main bank", "Other bank" and "Travel expense". The interfaces of an infotype can differ in appearance according to the subtype or can carry out other plausibility checks. For example, infotype "Family" has different screens for spouses and children.

2.4.3 SAP "Parties" Module

The SAP ERP does not contain a Parties module, the information regarding all parties is separated in the modules that were previously described. The "Party" information for an employee exists in the Personnel Administration sub-module within the Human Resources module. The Personal Data and Address infotypes provide a parallel to some of what is provided in the OFBiz Party module for employees. The following figure shows the screen used to edit the information of a given employee (infotype 002 – Personal data).

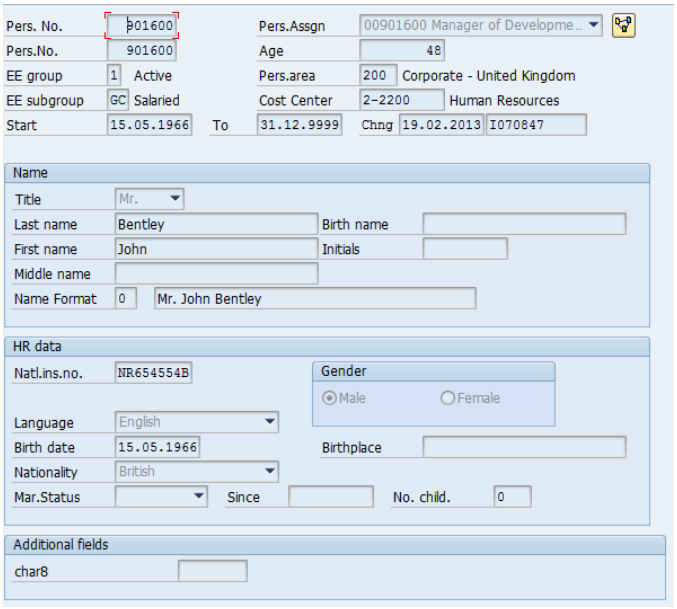


Figure 20 – Personal Data edition screen

In OFBiz the Parties module also defines, for example, customers whereas in SAP the customers are stored in the Sales & Distribution module. The following figure shows the Customer general data edition screen (KNA1 table) .

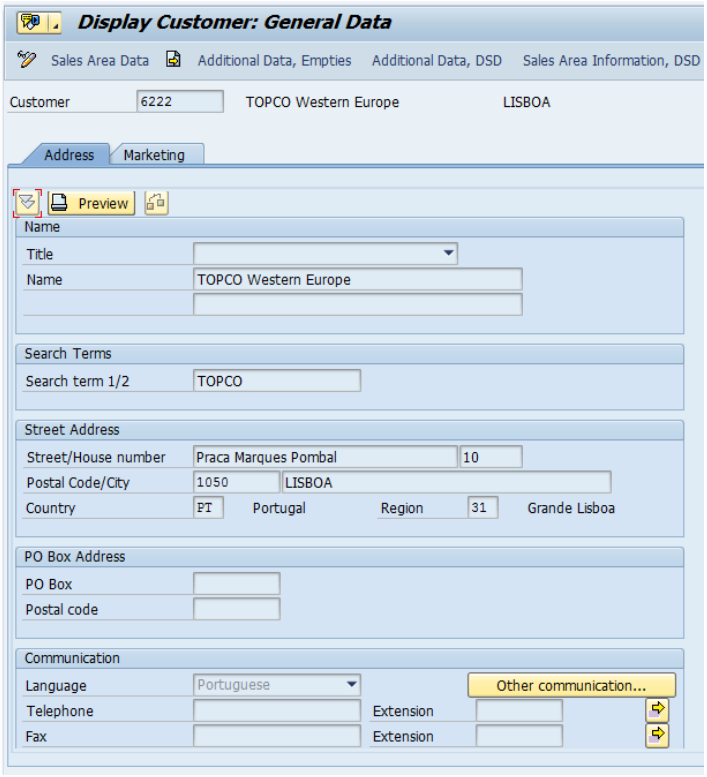


Figure 21 – Customer general data edition screen

Also the table that holds the records of the Customers have no relationship with the ones that store the Personal Data information in the Human Resources module, so they are clearly two distinct entities.

2.5 Summary

This chapter intended to explain what are the objectives of Enterprise Resource Planning software and what advantages they bring to the companies that use them. The first topic explains some of the concepts behind ERPs, such as business processes and functional areas.

The second topic enumerates and describes some of the more relevant functional areas that a company can have. By explaining the responsibilities of the functional areas we described some of the features that ERPs might have. Also in this topic, we saw that the responsibilities of a functional area usually require information of other functional areas, as such we identified some of the possible interactions between functional areas. This proves that having a system capable of sharing information between functional areas, and showing them to the right people at the right time can be helpful.

The next topic introduced the Party module, briefly explaining what are the objectives of that module. To better understand what entities the Party module interacts with, we explain each of the involved entities, why they are needed and how they related with each other by showing a generic Party data model. This data model is very similar to the one implemented by Apache OFBiz.

In the previous topic a small analysis of the SAP ERP is made. This analysis focuses on the main application modules giving a brief idea of what is covered by the application. The core concepts of the SAP ERP are presented, one of the most important concept is the infotype. Knowing what an infotype is, we show the edition screens for some of the entities that belong to the Parties module. By seeing that the model used by SAP does no match the one that we explained above, and seeing completely different screens and infotypes for maintaining employees and customers we concluded that SAP ERP does not have a Parties module.

The analysis of the data model that support SAP ERP and the generic data model that OFBiz uses allows us to answer the second literature question (LQ2). We can conclude that the data models used are completely different and that SAP does not have the concept of Party. Instead SAP separates the many entities that a Party encapsulate into distinct entities.

3 Apache Open for Business (OFBiz)

"*Apache OFBiz™ is an open source product for the automation of enterprise processes that includes framework components and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), E-Business / E-Commerce, SCM (Supply Chain Management), MRP (Manufacturing Resource Planning), MMS/EAM (Maintenance Management System/Enterprise Asset Management), POS (Point Of Sale).*" (The Apache Software Foundation, 2015)

OFBiz is composed by the following modules/applications:

- Account Payable Manager;
- Account Receivable Manager;
- Accounting Manager;
- Catalog Manager;
- Content Manager;
- Facility Manager;
- Human Resources;
- Manufacturing Manager;
- Marketing Manager;
- Order Manager;
- Party Manager;
- SFA Manager;
- Work Effort Manager.

3.1 Major Application Modules

This chapter wil describe the major application modules that are included in Apache OFBiz 13.07.

3.1.1 Common Data

The common data component is used to store data that is common between all application components. This data includes entities such as Geographic Boundaries, Units of Measure, Status Codes, Enumerations, etc... The majority of this data is created when the system is first installed and requires little change over time.

3.1.2 Content

The content management module allows users to store and organize information such as newsletters, enterprise documents or other web content. This content can be organized into trees, free-form association in graphs, lists, named maps, templates etc. The content module is also used to store information regarding the visits to the web store.

3.1.3 Security

The Security entities are used to manage login accounts as well as support the login audit infrastructure and the security permissions throughout the application.

3.1.4 Party

The Party module is used to maintain records of every entity with which the company interacts with, such as employees, customers, suppliers, etc. Each entity can store information regarding contact mechanisms (email, phone number, etc.) as well as communication events between the company and said entities. Parties are organized into Party groups and Party roles, grouping common entities in a group and defining the role of the entity within the company.

3.1.5 Product

The product module contains all the products that are used within the company, either sold or used to create other products. These products can be goods (raw materials, finished products, etc.) or services. The product module allows users to maintain all the information of those products in a centralized place, this information can include the price, the status, the location of the product, etc. Products are also organized into various categories and a product can be associated to one or multiple categories. Products can have a set of features associated with them, e.g. on a clothing store a shirt can have features like size and color. The product module also supports creating rules that can be applied to products under certain circumstances e.g.

discounts. Support for multiple currencies is also available as would be expected. OFBiz organizes product categories into catalogs, allowing users to define promotions and other specifications for different sales channels.

3.1.6 Order

The order module is used to store information about sales and purchase orders. The orders will be associated with a set of entities and can create tasks for employees to fulfill the order. The payment information can also be used to automate the invoice and payment of the order.

3.1.7 Facility

The facility module is used to manage all physical locations, e.g. warehouses, stores, or offices. Facilities can be used to specify where an employee works and/or where a given product is located. Facilities can have other parties associated with them such as: the organization unit that is responsible for the facility and the person who is the facility manager. Facility groups can be used to specify a hierarchy between facilities or group them into different geographical groups, e.g. country or city that can be used for marketing or pricing purposes.

3.1.8 Shipment

The shipment entities are used to keep track of incoming and outgoing shipments. This module is directly related with the inventory management since inventory items can arrive or be sent using a shipment carrier. Inventory items can be associated into Shipment Packages and be shipped using a Shipping Route.

3.1.9 Accounting

The Accounting entities are used to maintain the financial records of the company and all related Parties. "*The Accounting entities are organized according to age old and generally accepted principles such as double-entry accounting, a General Ledger with hierarchical accounts, journals and posting of transactions and corresponding entries"* (Jones, 2015). The accounting module supports the specification of custom fiscal periods.

3.1.10 Marketing

The Marketing entities are used to track information about Marketing Campaigns and related information such as Contact Lists (mailing, email, or calling lists) and Tracking Codes.

3.1.11 Work Effort

The work effort module can be used to track the work of employees, allowing users to create and manage projects, tasks, as well as simple to-do items or calendar items. All of these concepts can also be automatically created by the system when a given event happens. The work effort module also has support for keeping track of timesheets and pay rate of specific Parties performing Work Efforts.

3.1.12 Human Resources

The Human Resources entities are used to keep track of positions, responsibilities, skills, employment, termination, benefits, training, pay grades and payroll preferences, performance reviews, resumes and applications, and other Human Resources related information.

3.2 OFBiz technical overview

OFBiz is an open-source ERP implemented in Java. The application makes use of a lot of existing third-party components and frameworks, like Apache Tomcat[4] and Jaxen[5]. In this chapter we will describe the main application components of OFBiz, how they are implemented and how they work together. Let's start by taking a look at how the OFBiz source-code is organized. This technical overview was written based on the Apache OFBiz version 13.07.

The information contained in this chapter is an excerpt of the technical documentation that is included in the Appendix I.

3.2.1 OFBiz eclipse project

The eclipse project for OFBiz is composed of the following folders:

- applications – this folder contains the definition of the entities, services and user interface for each of the OFBiz application modules. This is where the core business logic is located;
- framework – this folder contains all the source code of the OFBiz framework;
- hot-deploy – this folder contains components that will be loaded after the OFBiz components (custom application modules should be created here);
- lib – contains the libraries used by OFBiz;
- runtime – this folder contains the logs and output files;
- specialpurpose – this folder contains special purpose built modules, like the e-commerce module;

[4] http://tomcat.apache.org/
[5] http://jaxen.org/

- themes – this folder contains the user interface themes that are common to all OFBiz application modules (like the main application bar, footer, cascading style sheets, etc.);
- tools – this folder contains command-line scripts used to start/stop OFBiz as well as manage SVN/Git folders.

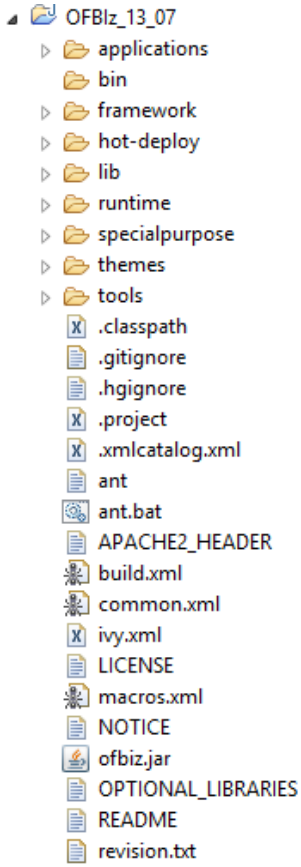


Figure 22 – OFBiz project structure.

3.2.1.1 Application module structure

Inside the applications folder there are sub-folders for each of the OFBiz modules. Each module can contain the following folders:

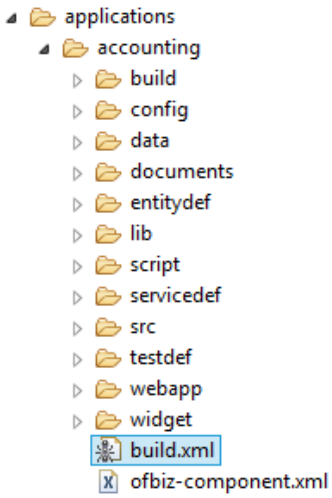


Figure 23 – OFBiz application module structure.

- build – contains the result of the compilation of the application module source-code;
- config – contains configuration files used within the module, e.g. application texts;
- data – contains seed, or test data files as well as help/documentation files;
- documents – contains the xml definition for help documents;
- entitydef – contains the entity definition files;
- lib – contains external libraries;
- script – contains mini-language scripts;
- servicedef – contains the service definitions;
- src – contains all Java source-code files;
- testdef – contains test scenarios;
- webapp – contains the web application itself. It contains the definition of actions and views;
- widget – contains the definition of the application views.

The ofbiz-component.xml file is used to let OFBiz know what is contained within the module and where it is located.

The build.xml file is used to instruct Apache Ant[6] how the application module should be compiled.

3.2.2 MVC implementation

OFBiz makes use of the MVC (Model-View-Controller) architectural pattern. The model is composed by a set of entities managed by an entity engine. The controller is implemented by a control servlet that will be explained later in detail. At last the views, are implemented as widgets. The following chapters will look in detail into how the MVC is implemented in OFBiz.

[6] http://ant.apache.org/

3.2.2.1 Entity Engine

The OFBiz entities are the base of the model in the MVC. Each entity is related to one or more database tables, e.g. the person entity defines the fields from the Person table where the information used to describe a person is contained. "*The Open For Business Entity Engine is a set of tools and patterns used to model and manage entity specific data*" (The Apache Software Foundation, 2015). The primary objective of the entity engine is to eliminate the need to develop entity/service specific persistence code. In order to achieve this level of abstraction, each entity must be configured in the entity engine, specifying the name of the entity, its fields, the type of each field and the relations with other entities. This data access layer abstraction prevents errors on persistence code by enforcing database rules specified in the configuration files and having a generic core capable of dealing with a wide variety of entities and database systems.

Entity Datasources

The entity datasources define, as the name says, the source of the data. By default, OFBiz will use a local Apache Derby[7] relational database management system. These configurations can be found in the following file: "/framework/entity/config/entityengine.xml". The following xml excerpt contains the default datasource definition:

```
<datasource name="localderby"
        helper-class="org.ofbiz.entity.datasource.GenericHelperDAO"
        schema-name="OFBIZ"
        field-type-name="derby"
        check-on-start="true"
        add-missing-on-start="true"
        use-pk-constraint-names="false"
        use-indices-unique="false"
        alias-view-columns="false"
        use-order-by-nulls="true"
        offset-style="fetch">
    <read-data reader-name="tenant"/>
    <read-data reader-name="seed"/>
    <read-data reader-name="seed-initial"/>
    <read-data reader-name="demo"/>
    <read-data reader-name="ext"/>
    <read-data reader-name="ext-test"/>
    <read-data reader-name="ext-demo"/>
    <inline-jdbc
            jdbc-driver="org.apache.derby.jdbc.EmbeddedDriver"
            jdbc-uri="jdbc:derby:ofbiz;create=true"
            jdbc-username="ofbiz"
            jdbc-password-lookup="derby-ofbiz"
            isolation-level="ReadCommitted"
            pool-minsize="2"
            pool-maxsize="250"
            time-between-eviction-runs-millis="600000"/>
</datasource>
```

[7] https://db.apache.org/derby/

The datasource element specifies, among other things, the JDBC driver, location, username and password as well as the schema name and isolation levels.

Entity Delegators

OFBiz uses entity delegators to access the database. The delegators provide means to execute CRUD operations to the database but do not execute these operations themselves, instead the delegator will look for the datasource of the entity group in which the entity being accessed is contained and delegate the operation to the specific datasource. The delegators are defined in the same file as the datasources. The following xml excerpt shows the default OFBiz delegator:

```
<delegator name="default" entity-model-reader="main"
      entity-group-reader="main" entity-eca-reader="main"
      distributed-cache-clear-enabled="false">
        <group-map group-name="org.ofbiz"
            datasource-name="localderby"/>
        <group-map group-name="org.ofbiz.olap"
            datasource-name="localderbyolap"/>
        <group-map group-name="org.ofbiz.tenant"
            datasource-name="localderbytenant"/>
</delegator>
```

Entity Groups

An entity group is, as the name indicates, a named group that contains multiple entities that are located on the same datasource. This name is used by the entity engine delegators to decide which datasource contains which entity. The main entity group of OFBiz is called “org.ofbiz”, by default all entities are associated with this group.

The association between entities and entity group are declared in the entitygroup*.xml file inside the entitydef folder under each component.

```
<entity-group group="org.ofbiz.olap"
      entity="SalesInvoiceItemFact"/>
```

Entity definition

In OFBiz there is no need to create new Java classes whenever we need to create a new database entity and use it in the application. To accomplish this, each entity is configured in a XML file that contains the fields, field types, primary keys, foreign keys, and so on, for each entity. Because there is no Java class associated with an entity, the framework needs to be smart and generic enough to support all possible entities, as such, the manipulation of an entity on the application side is made using generic objects such as generic maps. These generic maps provide access to all database fields by name.

The entities are configured in the /entitydef/entitymodel*.xml file within each application component. The following XML excerpt shows the definition of the Party entity.

```
<entity entity-name="Party" package-name="org.ofbiz.party.party"
  title="Party Entity">
  <field name="partyId" type="id-ne"></field>
  <field name="partyTypeId" type="id-ne"></field>
  <field name="externalId" type="id"></field>
  <field name="preferredCurrencyUomId" type="id-ne"></field>
  <field name="description" type="very-long"></field>
  <field name="statusId" type="id-ne"></field>
  <field name="createdDate" type="date-time"></field>
  <field name="createdByUserLogin" type="id-vlong"></field>
  <field name="lastModifiedDate" type="date-time"></field>
  <field name="lastModifiedByUserLogin" type="id-vlong"></field>
  <field name="dataSourceId" type="id"></field>
  <field name="isUnread" type="indicator"></field>
  <prim-key field="partyId" />
  <relation type="one" fk-name="PARTY_PTY_TYP"
      rel-entity-name="PartyType">
    <key-map field-name="partyTypeId" />
  </relation>
  <relation type="one" fk-name="PARTY_CUL" title="CreatedBy"
    rel-entity-name="UserLogin">
    <key-map field-name="createdByUserLogin"
      rel-field-name="userLoginId" />
  </relation>
  <relation type="one" fk-name="PARTY_LMCUL" title="LastModifiedBy"
    rel-entity-name="UserLogin">
    <key-map field-name="lastModifiedByUserLogin"
      rel-field-name="userLoginId" />
  </relation>
  <relation type="one" fk-name="PARTY_PREF_CRNCY"
    rel-entity-name="Uom">
    <key-map field-name="preferredCurrencyUomId"
      rel-field-name="uomId" />
  </relation>
  <relation type="one" fk-name="PARTY_STATUSITM"
    rel-entity-name="StatusItem">
    <key-map field-name="statusId" />
  </relation>
  <relation type="many" rel-entity-name="PartyTypeAttr">
    <key-map field-name="partyTypeId" />
  </relation>
  <relation type="one" fk-name="PARTY_DATSRC"
    rel-entity-name="DataSource">
    <key-map field-name="dataSourceId" />
  </relation>
  <index name="PARTYEXT_ID_IDX">
    <index-field name="externalId" />
  </index>
</entity>
```

View Entities

In addition to the OFBiz entities that map the fields of a single database table, OFBiz also supports View Entities. View Entities allow the creation of a “virtual” entity, composed by fields

of one or more database tables, similar to what can be achieved by using a “View” in Oracle Database[8] or Microsoft SQL Server[9].

The fields of a View entity will be alias of the original fields and can be either directly fetched from one of the view entities or calculated. View entities are defined in the same location as the entities and need to be associated with an entity group. The XML excerpt below shows the definition of the PartyAndGroup view entity.

```
<view-entity entity-name="PartyAndGroup"
      package-name="org.ofbiz.party.party"
      title="Party and Party Group View Entity">
      <member-entity entity-alias="PTY" entity-name="Party" />
      <member-entity entity-alias="PGRP"
            entity-name="PartyGroup" />
      <alias-all entity-alias="PTY" />
      <alias-all entity-alias="PGRP" />
      <view-link entity-alias="PTY" rel-entity-alias="PGRP">
            <key-map field-name="partyId" />
      </view-link>
</view-entity>
```

This view would generate an SQL as the one presented below:

```
SELECT
    PTY.PREFERRED_CURRENCY_UOM_ID,
    PTY.PARTY_TYPE_ID,
    PTY.EXTERNAL_ID,
    PTY.DATA_SOURCE_ID,
    PTY.IS_UNREAD,
    PTY.STATUS_ID,
    PTY.LAST_MODIFIED_BY_USER_LOGIN,
    PTY.PARTY_ID,
    PTY.DESCRIPTION,
    PTY.LAST_MODIFIED_DATE,
    PTY.CREATED_DATE,
    PTY.CREATED_BY_USER_LOGIN,
    PGRP.GROUP_NAME,
    PGRP.LOGO_IMAGE_URL,
    PGRP.GROUP_NAME_LOCAL,
    PGRP.TICKER_SYMBOL,
    PGRP.OFFICE_SITE_NAME,
    PGRP.ANNUAL_REVENUE,
    PGRP.NUM_EMPLOYEES,
    PGRP.COMMENTS
FROM OFBIZ.PARTY PTY
INNER JOIN OFBIZ.PARTY_GROUP PGRP ON PTY.PARTY_ID = PGRP.PARTY_ID
ORDER BY PTY.PARTY_ID ASC
```

The view definition presented above defines an entity that joins the Party entity with the PartyGroup entity and selects all the resulting fields. In this case, as both entities probably have

[8] https://www.oracle.com/database/index.html
[9] http://www.microsoft.com/en-us/server-cloud/products/sql-server/

fields with the same name we define the alias of “PTY” for the Party entity and “PGRP” for the PartyGroup entity. In short, the member-entity element defines what entities the view will use and what aliases they will have, the alias-all indicates that all fields from the entity with the provided alias will be selected and the view-link defines how the entities are combined, in this case they will be combined by the values of the field named “partyId”.

Extended Entities

Extending entities can be useful when we need to add a new field to an existing entity but do not want to change OFBiz original entity definitions (which would cause some problems when upgrading to newer versions). The following example extends the entity “Visitor” with the “partyId” field:

```
<extend-entity entity-name="Visitor">
  <field name="partyId" type="id"></field>
  <relation type="one" fk-name="VISITOR_PARTY"
      rel-entity-name="Party">
    <key-map field-name="partyId" />
  </relation>
</extend-entity>
```

Database Independence

OFBiz is database independent, with a few exceptions. As the data types of table fields vary from database to database, OFBiz has its logic attached to OFBiz data types that can be considered as “virtual” data types that will be converted to the database specific data type at runtime. The configuration of the specificities of each supported database system can be found under the /framework/entity/fieldtype/.

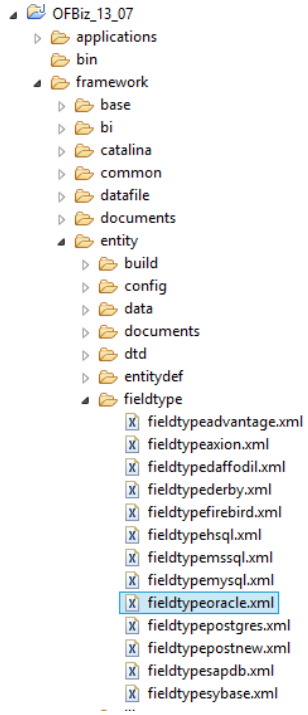


Figure 24 – Database field configuration.

Inside the fieldtype folder there is one XML file for each database system supported by OFBiz. Each file contains the specifications of each OFBiz type, database type and Java type. The XML displayed below shows the field type configuration for Oracle Database.

```
<field-type-def type="blob"
      sql-type="BLOB" java-type="java.sql.Blob"/>
<field-type-def type="byte-array"
      sql-type="BLOB" java-type="byte[]"/>
<field-type-def type="object"
      sql-type="BLOB" java-type="Object"/>
<field-type-def type="date-time"
      sql-type="TIMESTAMP" sql-type-alias="TIMESTAMP(6)"
      java-type="java.sql.Timestamp"/>
<field-type-def type="date"
      sql-type="DATE" java-type="java.sql.Date"/>
<field-type-def type="time"
      sql-type="DATE" java-type="java.sql.Time"/>
<field-type-def type="url"
      sql-type="VARCHAR2(255)" java-type="String"/>
<field-type-def type="id-ne"
      sql-type="VARCHAR2(20)" java-type="String"/>
<field-type-def type="id-long-ne"
      sql-type="VARCHAR2(60)" java-type="String"/>
```

3.2.2.2 The Control Servlet

The controller in OFBiz is called the control servlet. The control servlet is responsible for listening and responding to all incoming requests from the end-user. The control servlet acts a

connection between the entity engine, service engine and presentation layer. The control servlet implements the front controller pattern.

The following list enumerates a set of common tasks that are executed by the control servlet:

- Perform security checks;
- Log the request;
- Perform pre-processing;
- Lookup the defined processing for the request;
- Process events, if any;
- Determine the response and render it.

Each module of the OFBiz has its own control servlet defined in the web.xml file. The web.xml file can be broken into the following configurations:

- A name and description:

```
<display-name>Open For Business - Party Manager</display-name>
<description>Party Manager Module of the Open For Business
Project</description>
```

- What parameters will be available for the controllers, views and services to use:

```
<context-param>
  <param-name>entityDelegatorName</param-name>
  <param-value>default</param-value>
  <description>The Name of the Entity Delegator to use, defined in
entityengine.xml</description>
</context-param>
<context-param>
  <param-name>localDispatcherName</param-name>
  <param-value>partymgr</param-value>
  <description>A unique name used to identify/recognize the local
dispatcher for the Service Engine</description>
</context-param>
<context-param>
  <param-name>mainDecoratorLocation</param-name>
  <param-value>
      component://party/widget/partymgr/CommonScreens.xml
  </param-value>
  <description>The location of the main-decorator screen to use for
      this webapp; referred to as a context variable in screen def
      XML files.
  </description>
</context-param>
<context-param>
  <param-name>scriptLocationPath</param-name>
  <param-value>/WEB-INF/bsh</param-value>
  <description>BeanShell Script Location</description>
</context-param>
```

These parameters are called context parameters. Most of the application modules will configure at least the following parameters:

- Entity delegator – used for database access;
- Local service dispatcher – used to invoke services;
- Location of the main screen decorator – used to decorate all screens of the module;
- Location of BeanShell scripts. BeanShell scripts are currently deprecated in favor of Groovy scripts so this parameter is rarely used;

- What files can be served to users:

```
<filter>
  <filter-name>ContextFilter</filter-name>
  <display-name>ContextFilter</display-name>
  <filter-class>org.ofbiz.webapp.control.ContextFilter</filter-
class>
  <init-param>
    <param-name>disableContextSecurity</param-name>
    <param-value>N</param-value>
  </init-param>
  <init-param>
    <param-name>allowedPaths</param-name>
    <param-value>
      /error:/control:/select:/index.html:/index.jsp:/default.html:
      /default.jsp:/images:/static:/js
    </param-value>
  </init-param>
  <init-param>
    <param-name>errorCode</param-name>
    <param-value>403</param-value>
  </init-param>
  <init-param>
    <param-name>redirectPath</param-name>
    <param-value>/control/main</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>ContextFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

This is called context security filter and it's used to restrict access to certain application files. By default, the access to all files is rejected until it is specifically defined that a file or folder can be requested. The list of allowed files and folders is defined inside an init-param element whose param-name inner element value is "allowedPaths".

In the example above the following is permitted:

- Access to all files inside the following folders: /error, /control, /select, /images, /static, /js. Note that the "control" folder does not exist physically, but it is part of an expected URL pattern.
- Access to the /index.html, /index.jsp, /default.html, /default.jsp.

If a file that is not permitted is requested the server can either return an HTTP error with the error code specified in the filter parameter named "errorCode", or it can redirect to the page specified in the filter parameter named "redirectPath".

- Implementation of event listeners and control servlets:

```
<listener>
  <listener-
class>org.ofbiz.webapp.control.ControlEventListener</listener-
class>
</listener>
<listener>
  <listener-
class>org.ofbiz.webapp.control.LoginEventListener</listener-class>
</listener>
<servlet>
  <servlet-name>ControlServlet</servlet-name>
  <display-name>ControlServlet</display-name>
  <description>Main Control Servlet</description>
  <servlet-class>org.ofbiz.webapp.control.ControlServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

- What URLs should be intercepted by the control servlet:

```
<servlet-mapping>
  <servlet-name>ControlServlet</servlet-name>
  <url-pattern>/control/*</url-pattern>
</servlet-mapping>
```

In this case only requests that start with /control/ will be intercepted by OFBiz framework, this means that requests that do not match this pattern will be handled by the default server. OFBiz uses an embedded Apache Tomcat server by default. This servlet mapping is the requirement that led to the definition of the /control in the allowed paths of the context security filter.

The web.xml file explained above is only the entry point of the control servlet and does not contain the actual URLs that will be served and what will actually be done. This is specified in the controller.xml file. This file contains the requests that the module is capable of responding to, what security constraints must be met in order to execute the request logic, what events are fired, what “commands” need to be executed and what View should be written as output.

The possible elements within a controller element must appear in the following order:

- Include;
- Description;
- Handler;
- Request-map;
- View-map.

Controller configurations

OFBiz controllers are responsible for handling all OFBIz requests, it's what contains the logic of the web application. A controller defines what requests can be made, what responses can be sent to the client and who is responsible for building the response.

Handlers

Handlers are the entities responsible for building a certain type of response, e.g. a view (which will generate HTML). There are two types of handlers in OFBiz:

- View Handlers – Responsible for rendering a specific view to html;
- Event handlers – Responsible for triggering events.

The most commonly used view handlers are:

Handler name (s)	Handler class	Description
screen **screenxml** **screentext** **screencsv**	org.ofbiz.widget.screen.MacroScreenViewHandler	Common handler used to display OFBiz screens.
screenfop	org.ofbiz.widget.screen.ScreenFopViewHandler	Handler that uses XLS-FO formatted templates to generate PDF, PCL, PostScript, etc.
jsp	org.ofbiz.webapp.view.JspViewHandler	Handler for Java Server Pages.

Table 3 – OFBiz view handlers.

When a request is made and it needs to execute complex logic before deciding what should be sent to the client an event will be triggered. Events are used to run Java code, a Simple Method or a Groovy[10] script. The most commonly used event handlers are:

Handler name	Handler Class	Description
java	org.ofbiz.webapp.event.JavaEventHandler	Static Method Java Event Handler
soap	org.ofbiz.webapp.event.SOAPEventHandler	SOAP event handler.
xmlrpc	org.ofbiz.webapp.event.XmlRpcEventHandler	XML RPC event handler.
service	org.ofbiz.webapp.event.ServiceEventHandler	Handler used to invoke OFBiz services.
service-multi	org.ofbiz.webapp.event.ServiceMultiEventHandler	Event handler for running a service multiple times; for bulk forms

[10] http://www.groovy-lang.org/

Handler name	Handler Class	Description
service-stream	org.ofbiz.webapp.event.ServiceStreamHandler	Handler for services that need to access raw streams when receiving requests from the clients. The input and output streams are the only parameters.
simple	org.ofbiz.webapp.event.SimpleEventHandler	Simple event handler. Used to invoke events defined using Mini-Language.
groovy	org.ofbiz.webapp.event.GroovyEventHandler	Groovy event handler. Used to invoke events defined using Groovy.
rome	org.ofbiz.webapp.event.RomeEventHandler	Rome event handler. Used to invoke events defined using Rome e.g. RSS feeds.

Table 4 – OFBiz event handlers.

Request Maps

Request maps are responsible for specifying the actual logic to be executed when a request is made to a specific URL. The following XML is the definition for the action responsible for creating a new credit card.

```
<request-map uri="createCreditCard">
  <security https="true" auth="true" />
  <event type="simple"
      path="component://accounting/script/org/ofbiz/accounting/
      payment/PaymentMethodEvents.xml"
      invoke="createCreditCard" />
  <response name="success" type="view" value="viewprofile" />
  <response name="address" type="view" value="editcontactmech" />
  <response name="error" type="view" value="editcreditcard" />
</request-map>
```

To create a new credit card the user must send a request to the application module, in this case it's the party module, with the "createCreditCard" in the path, as follows:

Module	Required for URL pattern	Request
/party	/control	/createCreditCard

Table 5 – OFBiz request URL composition.

Events

Events contain the business logic of the request, like fetching records from the database for display, reading the user inputs and updating a specific record, etc. Usually the events will return a string or set a context value that will be used by OFBIz to determine the appropriate response. Some of the simplest business logic, like CRUD operations on the database can actually be specified on the view definition and do not require an event, as we will see later. Events are used when:

- The code needs to be executed independently from the response;
- The event will decide what response needs to be sent to the client;

Java Events

Java events are java methods that must obey to the following rules:

- must be a static method;
- must return a java.lang.String;
- must receive two parameters:
    - javax.servlet.http.HttpServletRequest;
    - javax.servlet.http.HttpServletResponse.

The HttpServletRequest parameter can be used to access data from the users request such as URL parameters, HTTP headers, cookies, among other things. The most common usage is to retrieve URL/post parameters which can be done using the "getParameter" method:

```
String productId = request.getParameter("PRODUCT_ID");
String productIdTo = request.getParameter("PRODUCT_ID_TO");
String productAssocTypeId =
request.getParameter("PRODUCT_ASSOC_TYPE_ID");
String fromDateStr = request.getParameter("FROM_DATE");
```

The return of a Java event is used to choose an appropriate response for the request. The returned String must match one of the values in the "name" attribute of a "response" element within the request map. Example:

```
public static Map callingServiceTwo(DispatchContext dctx, Map
context) {
  String userId = (String) context.get("userId");
  Map resultMap = null;
  if (userId.equals("10000")) {
    resultMap = ServiceUtil.returnSuccess("Welcome John!");
  } else {
    resultMap = ServiceUtil.returnError(
      "Welcome unknown user");
  }
  return resultMap;
}
```

The ServiceUtil class provides methods to return “success” or “error” values with messages. Alternatively, a string with the response can be returned.

If no event is present in the request map the default “success” response will be chosen, however, if an event is present but it returns a string that does not match any of the defined response names then no response will be provided (the end-user will see a blank page on the browser).

Responses

The response elements dictate what will be done with the result of the processing of the request. There are three types of possible responses:

- View response;
- Request response;
- Request-redirect response;
- Request-redirect-noparam response;

The view response will render an OFBiz view and send it as response.

```
<request-map uri="setCustomer">
  <security https="true" auth="true"/>
  <response name="success" type="view" value="custsetting"/>
</request-map>
```

The “request” response is used to fire a request to another action, leaving the response to the client up to the logic of the requested action.

```
<request-map uri="createCustomer">
  <security https="true" auth="true" />
  <event type="simple"
    path="component://order/script/org/ofbiz/order/
      customer/CustomerEvents.xml"
    invoke="createCustomer" />
  <response name="success" type="request" value="finalizeOrder" />
  <response name="error" type="view" value="custsetting" />
</request-map>
```

The “request-redirect” will send a redirect response to the end-user browser, which will cause it to fire a request to a specific URL.

```
<request-map uri="makeQuickReturn">
  <security https="true" auth="true" />
  <event type="service-multi"
      invoke="createReturnAndItemOrAdjustment" />
  <response name="success" type="request-redirect"
      value="returnItems">
    <redirect-parameter name="returnId" />
  </response>
  <response name="error" type="view" value="orderview" />
</request-map>
```

The request-redirect-noparam response is similar to the request-redirect response but the new request won't have any of the parameters sent in the first request.

View Maps

As shown above, one of the possible return types for a request is a view. Whenever an action specifies a return of type "view" the value specified should be the name of an existing view map.

```
<request-map uri="main">
  <security https="true" auth="true" />
  <response name="success" type="view" value="main" />
</request-map>
<view-map name="main" type="screen"
 page="component://reports/widget/reports/CommonScreens.xml#main"/>
```

3.2.2.3 Screen Widgets

The "View" part of the MVC implementation in OFBiz is composed of Screen Widgets. Every view is a screen widget or is contained within one. Views are responsible to present information to the end-user as well as providing the means to change it.

The screen widgets are defined per application module thus they are contained within an XML file within the "widgets" folder of a given OFBiz module. To be able to use screen widgets the controller must be aware of what class is responsible for handling the views, the default handler is the "org.ofbiz.widget.screen.MacroScreenViewHandler".

```
<handler name="screen" type="view"
  class="org.ofbiz.widget.screen.MacroScreenViewHandler" />
```

A screen is defined by using the "screen" XML element and a given name. Within the screen element a section needs to be defined (one section per screen). Sections act as containers for the view elements and can contain logic to decide whether or not to show a given element. This can be accomplished by using the If-then-else structure of the section element. The implementation of conditional screens can be achieved by using the "condition" element, which represents the "if" part of the if-then-else structure.

The following conditions are available in OFBiz:

- and
- or
- xor
- not
- if-service-permission
- if-has-permission
- if-entity-permission
- if-validate-method
- if-compare
- if-compare-field
- if-regexp
- if-empty
- if-empty-section

If the condition evaluates to true the “then” part will be executed, as expected. In the screen widgets, the “then” part is composed by two elements: the “actions” and the “widgets”. When the condition is evaluated to false, only the “fail-widgets” element is executed, meaning that the content within the “actions” and “widgets” elements will be skipped.

The “actions” element is where the logic to retrieve elements from the database, manipulation or transformation of input parameters should be contained. When communicating with the database only “read” operations are allowed here. The widget actions can be categorized into three distinct groups:

- The actions that retrieve values from the database and place it in variables;
- The actions that invoke scripts or services;
- The actions that manipulate variables.

Database retrieval actions

- entity-and – Used to find an entity by specifying a set of filters that will be concatenated by an “and” operator;
- entity-condition – The entity condition element is used to find entities by some specific criteria;
- entity-one – The entity-one element is used to find an entity by its primary-key;
- get-related-one – The get-related-one element is used to retrieve an entity that is related to another entity based on the declared relations of the primary entity;
- get-related – The get-related element is similar to the get-related-one as it is also based on the declared relations of an entity but instead of retrieving only one value it is used to retrieve a list of related entities (e.g. entities that have N records related to 1 entity).

Scripts and service invocation actions

- script – used to invoke a script (e.g. a Groovy script);
- call-service – used to invoke an OFBiz service.

Variable manipulation actions

- property-map – The property-map element provides access to OFBiz text resources by mapping all properties of a given resource to a specified map variable;
- property-to-field – The property-to-field element is used to retrieve the value of a single property from an OFBiz resource;
- set – used to declare a variable or set the value of an existing variable.

A detailed description of these elements can be found in the Mini-Language chapter.

The widgets element acts as a container for the elements that will be displayed, like images, labels, links, buttons, etc.

Context Variables

Screen widgets have access to a pre-defined set of OFBiz objects. These objects can be used to access HTTP request parameters, obtain information regarding the user login, etc. The list of available context variables is listed in the Mini-Language chapter.

Defining a screen widget

Screen widgets define what HTML will be written to the response for a given request. That said, the controller must have a request map that returns a view. Views are placed within an XML file whose name should end with "Screens" (e.g. PartyScreens, OrderScreens, etc.) and should be placed within the widget folder of the given module (e.g. applications/party/widget /partymgr/PartyScreens.xml). Assume that the PartyScreens.xml has a view called "findparty", this view could be referenced in a request map by adding "#<name of the view>" after the location of the file where it is located, e.g.

```
<view-map name="main" type="screen"
  page="component://party/widget/partymgr/
      PartyScreens.xml#findparty"/>
```

As said above, screen widgets are declared using the screen element that must contain at least one section element and one widgets element within the section. The first section element is required for a screen element and can be understood as the body for the page that will be rendered. There must be exactly one section per screen element.

The widgets element is where the content of the screen is placed. The widgets element supports multiple elements to render images, labels, links, etc.

The following XML illustrates the simplest possible OFBiz screen widget, containing only one label.

```
<screen name="SimplestScreen">
  <section>
    <widgets>
      <label text="Simplest Screen possible in OFBiz!" />
    </widgets>
  </section>
</screen>
```

A screen section can have conditions and actions, used for example to show or hide some content based on the value of a variable or request parameter. These can be achieved using the “conditions” and “actions” element.

The conditions element acts as an “If” and if the result of the conditions element evaluates to true the then part of the screen, composed of the actions and widgets elements, is executed otherwise the else part, composed only of the “fail-widgets” element is executed.

The next XML shows a screen that receives a request parameter called “show” and displays different labels based on the value of that parameter.

```
<screen name="ConditionalScreen">
  <section>
    <condition>
      <if-compare field="parameters.show" operator="equals"
        value="all" />
    </condition>
    <actions>
      <set field="showing" value="all" />
    </actions>
    <widgets>
      <label
        text="Condition passed. Showing widgets element.
             Showing: ${showing}" />
    </widgets>
    <fail-widgets>
      <section>
        <actions>
          <set field="showing" value="none" />
        </actions>
        <widgets>
          <label
            text="Condition failed! Showing fail-widgets element.
                  Showing: is: ${showing}" />
        </widgets>
      </section>
    </fail-widgets>
  </section>
</screen>
```

Screens can be reused within other screens by using the include-screen element, as shown below:

```
<screen name="NestedScreen">
  <section>
    <widgets>
      <label text="This is the simple screen content:" />
        <include-screen name="SimplestScreen" />
    </widgets>
  </section>
</screen>
```

Screen decorators

OFBiz uses the screen decorator pattern to render most of its screens. This helps to centralize the logic to render the application header and footer in one place and leave the actual content to be declared in each screen definition. The following XML shows the definition for the header and footer screens.

```
<screen name="header">
  <section>
    <widgets>
      <label text="This is the header"></label>
    </widgets>
  </section>
</screen>
<screen name="footer">
  <section>
    <widgets>
      <label text="This is the footer"></label>
    </widgets>
  </section>
</screen>
```

To use these screens in the simplest screen shown above we could simply include the screens but this is not the best approach. What is done in OFBiz instead is that a screen decorator will be created and that screen decorator will have one or many placeholders that the screens can customize, in this case only one will be used and it will be named “body”. This placeholder is defined using the decorator-section-include element and providing a name.

```
<screen name="simple-decorator">
  <section>
    <widgets>
      <include-screen name="header" />
      <decorator-section-include name="body" />
      <include-screen name="footer" />
    </widgets>
  </section>
</screen>
```

The following screen will use the new decorator (using the decorator-screen element) and specify the content for the “body” section (using the decorator-section element with the correct name) with the contents of the simplest screen show above.

```
<screen name="CompoundedScreen">
  <section>
    <widgets>
      <decorator-screen name="simple-decorator">
        <decorator-section name="body">
          <include-screen name="SimplestScreen" />
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

Forms

One of the required element of any application is the existence of forms, and OFBiz is no different. Forms are supported by declaring them in XML. Each module has its own forms file that should be named "<Module>Forms.xml" and placed in the widget folder of that module.

Forms are specified using the form element. The form element has the following attributes

- Name – specifies the name of the form;
- Type – specifies the type of the form. Can have the following values:
  - Single
  - List
  - Multi
  - Upload
- Target – name of the action that will receive the post request with the form values.

The form element can have several field elements, one for each input of the form. Each field also has a sub-element that specifies the type of the input field, e.g. text, dropdown, date, etc.

The following xml shows a simple form:

```
<forms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/
      widget-form.xsd">
  <form name="ReportSaftGenerationForm"
    type="single" target="GenerateSaft">
    <field name="dateBegin">
      <date-time />
    </field>
    <field name="dateEnd">
      <date-time />
    </field>
    <field name="submit">
      <submit />
    </field>
  </form>
</forms>
```

3.2.3 Service Engine

OFBiz contains a service layer that can be used from the web application, invoked from the outside via HTTP (using SOAP) or scheduled to run in the background at a specified time. OFBiz services can be of many different types: Simple or Java. When using Java, the service will be a static method with the input and output parameters passed in using a Map so they can be serialized and transmitted using SOAP via HTTP, if needed.

Each service must specify the name of the service engine in its definition, this service engine name is used by the Service Dispatcher to choose the appropriate service engine to use. OFBiz has one service dispatcher for each entity delegator. Apart from the language used to code the

service itself there must be a definition for the service. This definition indicates OFBiz, among other things, what engine should be used and what parameters the service is expecting. Services are declared using XML via the service configuration file. Since services are identified by its name, defining two services with the same name might cause the first service to be overridden by the second.

3.2.3.1 Service definition

Services are declared in XML files that are usually named services.xml and placed in the servicedef folder of the OFBiz module. The services.xml file starts with the “services” element and inside each service is declared using the “service” element.

```
<service name="deleteParty" engine="java"
  location="org.ofbiz.party.party.PartyServices"
  invoke="deleteParty"
  auth="true">
  <description>Delete a Party</description>
  <attribute name="partyId" type="String"
    mode="IN" optional="true" />
</service>
```

3.2.3.2 Mini-Language Services

OFBiz has many types of different service languages, one of them is the Mini-Language, referred to as Mini-Lang from now on. Services or events created using Mini-Lang are commonly referred to as simple methods. These simple methods have their logic written in XML. The following example shows a simple service that is used to delete a party role.

```
<service name="deletePartyRole" engine="simple"
    location="component://party/script/org/
      ofbiz/party/party/PartyServices.xml"
      invoke="deletePartyRole" auth="true">
  <description>Delete a Party Role (remove a Role from a Party).
    The logged in user must have PARTYMGR_DELETE or have
    permission to change the role of this partyId</description>
  <permission-service service-name="partyRolePermissionCheck"
      main-action="DELETE"/>
  <attribute name="partyId" type="String" mode="IN"
      optional="true"/>
  <attribute name="roleTypeId" type="String" mode="IN"
      optional="false"/>
</service>
```

```
<simple-method method-name="deletePartyRole" short-
      description="Delete a PartyRole">
  <entity-one entity-name="PartyRole" value-field="partyRole"/>
  <remove-value value-field="partyRole"/>
</simple-method>
```

All the elements supported by Mini-Lang are documented in the Mini-Language reference documentation and can be found here:

https://cwiki.apache.org/confluence/display/OFBADMIN/Mini+Language+-+minilang+-+simple-method+-+Reference

3.2.3.3 Java Services

Java services are java static methods that are used to code complex business rules that would be otherwise too difficult to maintain using the simple methods written in OFBiz Mini-Lang. These methods receive two parameters:

- the dispatch context, used to retrieve context variables like the datasource delegator;
- a map containing the service parameters.

The Java source-code, used to define services and/or events should be placed within the src folder.

```
public static Map<String, Object> createPerson(
      DispatchContext ctx, Map<String, ? extends Object> context) {

    Map<String, Object> result = FastMap.newInstance();
    Delegator delegator = ctx.getDelegator();
    Locale locale = (Locale) context.get("locale");
    GenericValue userLogin =
      (GenericValue) context.get("userLogin");

    String partyId = (String) context.get("partyId");
    String description = (String) context.get("description");

    //...

    result.put("partyId", partyId);
    result.put(
      ModelService.RESPONSE_MESSAGE,
      ModelService.RESPOND_SUCCESS);
    return result;
}
```

Service attributes

As explained above, the service parameters are defined in the XML file. Service parameters are named service attributes, the following shows the attributes for a service that receives the first name and last name and returns the full name.

```
<attribute name="firstName" type="String" mode="IN"
  optional="true"/>
<attribute name="lastName" type="String" mode="IN"
  optional="true"/>
<attribute name="fullName" type="String" mode="OUT"
  optional="true"/>
```

In the example above all attributes are defined as optional inputs but we could define required attributes by simply changing the value of the “optional” attribute to false. If the required attributes are not provided, the service is not executed and an error message will be displayed.

The input attributes can be accessed in the service by using the Map variable received as a parameter of the method, in this case the parameter is called context. The output attributes should be placed in a Map variable with the expected attribute name. The following example shows the Java code of the service explained above:

```
public static Map handleParameters(
      DispatchContext dctx, Map context) {

      String firstName = (String) context.get("firstName");
      String lastName = (String) context.get("lastName");
      String message = "firstName: " + firstName;
      message = message + "lastName: " + lastName;
      Map resultMap = ServiceUtil.returnSuccess(message);
      resultMap.put("fullName", firstName + " " + lastName);
      return resultMap;
}
```

All services have a few special attributes that are not explicitly declared but they exist. The attributes are:

- userLogin
- locale

The userLogin attribute contains information regarding the authenticated user that is required for permission checks. The locale defines the culture that should be used to retrieve application texts, format dates and currencies.

Service security

By using the context variables shown above services can execute authorization validations and return an error if the user doesn't have the required permissions. Note that these permissions can also be defined in the XML file. Services should only be used for authorization validation when the rules are too complex to define in XML.

```
public static Map serviceWithAuth(
      DispatchContext dctx, Map context) {
  Security security = dctx.getSecurity();
  Map resultMap = null;
  if (context.get("userLogin") == null
    || !security.hasPermission("TEST_VIEW",
      (GenericValue) context.get("userLogin"))) {
      resultMap = ServiceUtil
        .returnError("You have no access!");
  } else {
    resultMap = ServiceUtil.returnSuccess("Welcome!");
  }
  return resultMap;
}
```

Invoking other services

Access to other services is provided by the Local Dispatcher. To invoke a service, we simply call the "runSync" method providing the service name and a Java Map containing the input attributes.

```
public static Map callingServiceOne(
      DispatchContext dctx, Map context) {

  LocalDispatcher dispatcher = dctx.getDispatcher();
  Map resultMap = null;
  GenericValue userLogin = (GenericValue) context.get("userLogin");
  Locale locale = (Locale) context.get("locale");

  Map serviceTwoCtx = UtilMisc.toMap("firstName",
      "Jorge", "lastName", "Almeida", "userLogin",
      userLogin, "locale", locale);

  try {
    resultMap = dispatcher.runSync("ServiceTwo", serviceTwoCtx);
  } catch (GenericServiceException e) {
    Debug.logError(e, "Learning");
  }

  return resultMap;
}
```

There are three methods that can be used to invoke a service:

- runSync – runs the service synchronously and returns the result as a java Map;
- runSyncIgnore – runs the service synchronously and ignores the result. Nothing is returned;
- runAsync – runs the service asynchronously and nothing is returned.

```
<service name="sendOrderConfirmation" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderConfirmNotification">
  <description>Send a order confirmation</description>
  <implements service="orderNotificationInterface" />
</service>

<service name="sendOrderChangeNotification" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderChangeNotification">
  <description>Send a order notification</description>
  <implements service="orderNotificationInterface" />
</service>
```

3.2.3.4 Service Groups

Service groups are used to simplify the invocation of multiple services that must be executed in a specific order. Service groups define the set of services that are executed and the name of the service group. Service Groups can be invoked in the same way as a normal service. If the result-

-to-context attribute is set to true, the result of the service will be placed in the context for the other services to use.

```
<group name="createCreditCardAndAddress" send-mode="all">
  <invoke name="createPostalAddress" mode="sync"
    result-to-context="true" />
  <invoke name="createCreditCard" mode="sync" />
  <invoke name="createPartyContactMech" mode="sync" />
</group>
```

```
<request-map uri="createCreditCardAndPostalAddress">
  <security https="true" auth="true" />
  <event type="service" path=""
      invoke="createCreditCardAndAddress" />
  <response name="success" type="request" value="finalizeOrder" />
  <response name="error" type="view" value="billsetting" />
</request-map>
```

3.2.4 Internationalization

As would be expected OFBiz supports multiple languages for its user-interfaces. The User interface texts are defined in XML files usually stored in the config folder within each component, as shown below:

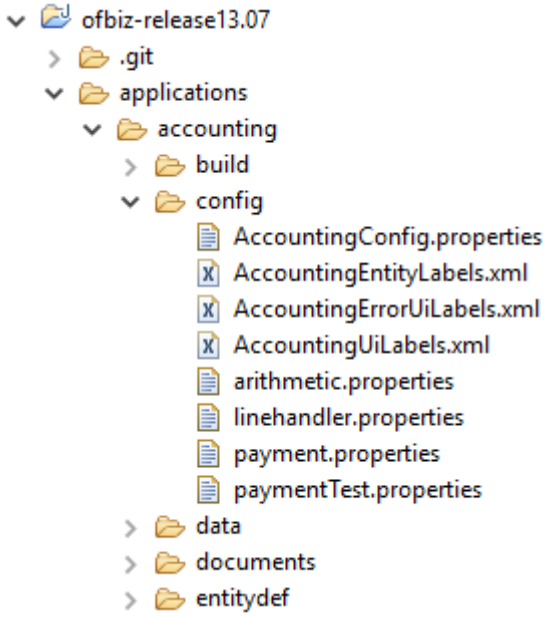


Figure 25 – OFBiz internationalization file structure.

The common application texts that can be shared between components are defined in the framework/common/config folder. The texts are identified by a key that has a value for each of the supported languages.

The following XML excerpt shows the definition of the possible values for the “CommonSave” property, which defines the common value for the “Save” text:

```
<property key="CommonSave">
   <value xml:lang="ar">حفظ</value>
   <value xml:lang="cs">Uložit</value>
   <value xml:lang="de">Speichern</value>
   <value xml:lang="en">Save</value>
   <value xml:lang="es">Guardar</value>
   <value xml:lang="fr">Enregistrer</value>
   <value xml:lang="hi-IN">□□□□□□□□□ □□□</value>
   <value xml:lang="it">Salva</value>
   <value xml:lang="ja">保存</value>
   <value xml:lang="nl">Opslaan</value>
   <value xml:lang="pt">Guardar</value>
   <value xml:lang="pt-BR">Salvar</value>
   <value xml:lang="ro">Salveaza</value>
   <value xml:lang="ru">Сохранить</value>
   <value xml:lang="th">บันทึก</value>
   <value xml:lang="vi">Luu</value>
   <value xml:lang="zh">保存</value>
   <value xml:lang="zh-CN">保存</value>
   <value xml:lang="zh-TW">儲存</value>
</property>
```

The following rules are applied when choosing the correct value to display:

- When the requested language is a country specification (e.g. Portuguese of Portugal) and a value is available for that specification then the value of the specification will be displayed;
- When the requested language is a country specification and only the non-specific language is available then the value for the non-specific language will be displayed;
- When the requested language is not defined but English is defined, the English value will be displayed;
- When no language is defined for the property the name of the property will be displayed;

3.2.4.1 Using texts in Mini-Lang

The property-map element provides access to OFBiz text resources by mapping all properties of a given resource to a specified map variable. This variable can then be used within a set element to retrieve the value of an existing property to another variable, e.g.:

```
<screen name="main-decorator">
  <section>
    <actions>
      <property-map resource="PartyUiLabels" map-name="uiLabelMap"
        global="true" />
      <set field="layoutSettings.companyName"
        from-field="uiLabelMap.PartyCompanyName"
        global="true" />
```

3.2.4.2 Using texts in Groovy

Accessing OFBiz text resources in a Groovy script is as simple as invoking the getResourceBundleMap from the available UtilProperties java class. The following code creates a map with all properties from the "PartyUiLabels" resource and then sets the variable "label" with the value of the PartyProfile resource property.

```
uiLabelMap = UtilProperties.getResourceBundleMap("PartyUiLabels",
      locale);
label = uiLabelMap.PartyProfile;
```

Alternatively, if only one property value is needed from the resource the getMessage method can be used instead.

```
UtilProperties.getMessage(resource,
      "PartyCannotLinkPartyToItSelf", locale)
```

3.2.4.3 Using texts in Java

Accessing text resources in Java events or services is done in the same way as it was described in the Groovy section since Groovy scripts makes use of Java classes.

3.2.5 Mini-Language

Mini-Lang is an XML defined script that is parsed and executed by the OFBiz framework. Its goal is to make repetitive tasks, like CRUD operations, simpler to develop. Since the logic is defined in XML it is somewhat easier to understand. Mini-Lang also has other advantages when compared to defining logic in Java besides being easier to write and understand. Methods defined using Mini- Lang do not need to be compiled, meaning that we can change the logic of our code when the website is running and a simple page-refresh is enough for our new code to be executed. One of the disadvantages of Mini-Lang is the debug capability, since it is defined using XML it does not support the line-by-line debug that is commonly used in Java. Taking into account this advantages and disadvantages, Mini-Lang is commonly used for simple services and events because even though Mini-Lang could be used to code complex services, the time that could take to debug the service, as well as understanding it, would exceed the time to develop it in Java.

Mini-Lang is used in OFBiz for three different purposes:

- Defining service logic;
- Defining event logic;
- Preparing data for screen widgets;

Part of the simplicity of using Mini-Lang comes from the fact that there is no need to declare variables, for example, if we need to set the value of a given variable we would simply write the following XML:

```
<set field="currentOrganizationPartyId" from-
      field="parameters.organizationPartyId" />
```

The above XML sets the field “currentOrganizationPartyId” with the value from the field “organizationPartyId” of the context parameters object. Alternatively, it is possible to set the variable with a specific value:

```
<set field="containsEmptyFields" type="Boolean" value="true" />
```

If the variable was already declared, then its value will be changed, if not, it will be declared and given the specified value.

3.2.5.1 Context Variables

In Mini-Lang we also have access to a set of “context” variables, e.g. service parameters. The following list enumerates the available context parameters. Some of the following parameters are only available on screen widgets and events.

- globalContext
  - is an instance of org.ofbiz.base.util.collections.MapStack
  - This is a context variable that can be used within all nested screens
- nullField
  - is an instance of org.ofbiz.entity.GenericEntity$NullField
  - This object is used to check for null values in database fields.
- availableLocales
  - is an instance of java.util.List
  - Contains the list of available locales supported
- locale
  - is an instance of java.util.Locale
  - Contains the current locale. This value is English by default and can be changed at any time. The current locale is the one that will be used to retrieve application texts.
- delegator
  - is an instance of org.ofbiz.entity.GenericDelegator
  - Used for communicating with the database.
- dispatcher
  - is an instance of org.ofbiz.service.GenericDispatcher

    - Used to call OFBiz services.
- security
    - is an instance of org.ofbiz.security.OFBizSecurity
    - Helper object that provides a number of methods useful to execute permission checks and other security related tasks.
- userLogin
    - is an instance of org.ofbiz.entity.GenericValue
    - Used to retrieve information about the logged in user.
    - This object is not always available since the user is not required to be logged in to visit a page, e.g. the e-commerce module.
- parameters
    - is an instance of java.util.Map
    - Is a key-value pair collection that contains the service/request parameters, request attributes, session attributes and ContextServlet attributes.

3.2.5.2 Creating a Simple Service

Services defined using Mini-Lang should be placed in a <Module>Services.xml file within the scripts folder of the module.

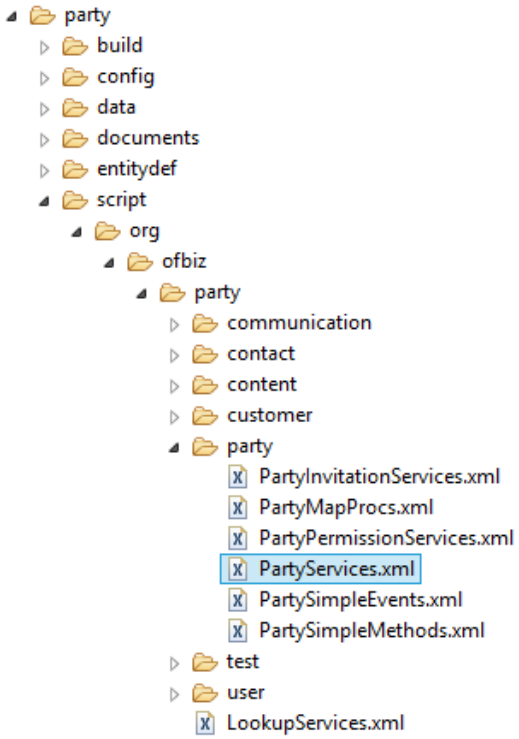


Figure 26 – OFBiz service definition file location.

The following XML defines a simple service that is used to create a new Party Role.

```
<simple-method method-name="createPartyRole"
  short-description="Create Party Role">
  <entity-one entity-name="PartyRole" value-field="partyRole" />
  <if-empty field="partyRole">
    <make-value entity-name="PartyRole" value-field="newEntity" />
    <set-pk-fields map="parameters" value-field="newEntity" />
    <create-value value-field="newEntity" />
  </if-empty>
</simple-method>
```

The “entity-one” element will select a PartyRole entity with the primary key specified in the partyRole field. If no entity is found a new one will be created. The “make-value” element creates a new PartyRole entity in the newEntity variable. The “set-pk-fields” will map the fields from the “parameters” context variable to the newEntity variable and then the newly created entity will be stored in the database using the “create-value” element.

The simple method needs to be registered as a service in the services.xml file, as follows:

```
<service name="createPartyRole" engine="simple"
  location="component://party/script/org/ofbiz/
    party/party/PartyServices.xml"
  invoke="createPartyRole" auth="true">
  <description>Create a Party Role (add a Role to a Party).
    The logged in user must have PARTYMGR_CREATE or have
    permission to change the role of this partyId</description>
  <permission-service service-name="partyRolePermissionCheck"
    main-action="CREATE" />
  <attribute name="partyId" type="String" mode="IN"
    optional="true" />
  <attribute name="roleTypeId" type="String" mode="IN"
    optional="false" />
</service>
```

The service can then be invoked from the controller, as follows:

```
<request-map uri="addrole">
  <security https="true" auth="true" />
  <event type="service" path="" invoke="createPartyRole" />
  <response name="success" type="view" value="viewprofile" />
  <response name="error" type="view" value="viewprofile" />
</request-map>
```

3.2.5.3 Creating a Simple Event

Simple methods used as simple events should be placed a <Module>Events.xml file within the scripts folder of the module.

The following simple method will only set the value for the parameters of the simple method “createUser” that will then be invoked using the “call-simple-method” element.

```
<simple-method method-name="createCustomer"
  short-description="Create Customer" login-required="false">
  <set field="require_email" value="true" />
  <set field="require_phone" value="false" />
  <set field="require_login" value="true" />
  <set field="create_allow_password" value="true" />
  <set field="parameters.roleTypeId" value="CUSTOMER" />
  <call-simple-method method-name="createUser" />
</simple-method>
```

The simple method can be invoked from the controller as follows:

```
<request-map uri="createCustomer">
  <security https="true" auth="true" />
  <event type="simple"
    path="component://party/script/org/ofbiz/
      party/user/UserEvents.xml"
    invoke="createCustomer" />
  <response name="success" type="view" value="viewprofile" />
  <response name="error" type="view" value="NewCustomer" />
</request-map>
```

3.3 Summary

This chapter provides an analysis of Apache OFBiz, an open-source ERP software. The first topic describes the major application modules that can be related to the functional areas shown in the previous chapter. The description for the application modules enumerate some of the features that are implemented by the software.

The second topic is dedicated to analyzing OFBiz from a technical point of view. We start by looking at how the source-code is structured and what are the components that can be included within an application module, since each application module has its own source-code folder. This topic covers how the MVC pattern is implemented in OFBiz, how the HTTP/HTTPS requests to the server are processed and how a response is returned to the user. We also show how OFBiz supports localized user interfaces. A localized user interface can vary the texts that are displayed depending on the culture of the user's browser. The integration between Web components and the background job infrastructure is also explained.

The contents presented in this chapter present enough information to specify what functional areas an ERP should cover and some of the features that it should have. This allows us to answer the first literature question (LQ1). With the analysis presented in this chapter and the previous one we can also answer the second research question (RQ2). We can conclude that an ERP should cover a high number of functional areas and business processes and that the commercial solution provides a greater number of features.

4 Customizing Apache OFBiz

One of the goals of this document is to provide guidance when choosing between customizing an open-source ERP such as Apache OFBiz or developing a new application with specific requirements in mind. To be able to do so, we will evaluate the effort of extending OFBiz by implementing a common requirement for Portuguese businesses, the generation of SAF-T (Portuguese version) audit files. The implemented SAF-T audit file generator will only be a prototype implementation as generating the whole audit file requires information from modules outside of the Parties module.

4.1 SAF-T (Portuguese version) audit file overview

The SAF-T audit file is a standardized XML file used for exporting accounting information of a company to the tax authorities or external auditors. The standard is defined by the Organisation for Economic Co-operation and Development (OECD). The SAF-T is a XML file with the objective of allowing tax authorities to retrieve fiscal data for companies in a digital format.

The XML file can be generated by applications with distinct scopes and as such the SAF-T file can have different sets of data, e.g. for both accounting and billing files should have the following information:

- Header
- Customers
- Tax information
- Payments

For accounting applications, the file should also contain the following information:

- General Ledgers

- Suppliers
- General Ledger entries

For billing applications, the file should contain the following information:

- Suppliers
- Products
- Sales invoices
- Movement of goods
- Working documents

For integrated applications file may also contain the following information, when available:

- General Ledgers
- Suppliers
- General Ledger entries
- Products
- Sales invoices
- Movement of goods
- Working documents
- Payments

Each set of information listed above will represent a table in the XML file. The SAF-T audit file expects a rigid structure for each type of information listed above, for example, the Header and Customers data should export the following information:

Index	Required	Field name	Notes	Format
1.1	Yes	Audit file (AuditFileVersion)	Specifies the version of the XML file. Different versions can be found in the Portuguese government website: http://www.portaldasfinancas.gov.pt.	Text, 10
1.2	Yes	Identification of the company (CompanyID)	The legal identification number of the company.	Text, 50
1.3	Yes	Fiscal identification number (TaxRegistrationNumber)	Portuguese NIF.	Integer, 9
1.4	Yes	Accounting system basis (TaxAccountingBasis)	Can have one of the following values: “C” – accounting, “E” – billing issued by third-parties “F” – billing “I” – integrated accounting and billing “P” – partial billing “R” – receipts “S” – automatic billing “T” – transport document	Text, 1
1.5	Yes	Company name (CompanyName)		Text, 100

1.6	No	Business name (BusinessName)		Text, 60
1.7	Yes	Company Address (CompanyAddress)		N/A
1.7.1	No	Building number (BuildingNumber)		Text, 10
1.7.2	No	Street name (StreetName)		Text, 90
1.7.3	Yes	Detailed address (AddressDetail)	Should include the street name, building number and floor, if applicable.	Text, 100
1.7.4	Yes	City (City)		Text, 50
1.7.5	Yes	Postal code (PostalCode)		Text, 8
1.7.6	No	Region (Region)		Text, 50
1.7.7	Yes	Country (Country)	Should be “PT”	Text, 2
1.8	Yes	Fiscal year (FiscalYear)	Should obey to IRC code rules if the fiscal year does not match the civil year, e.g. If the fiscal year starts at 01-03-2012 and ends in 01-03-2013 the fiscal year is 2012.	Integer, 4
1.9	Yes	Start date for the file records (StartDate)		Date
1.10	Yes	End date for the file records (EndDate)		Date
1.11	Yes	Currency Code (CurrencyCode)	Should be “EUR”	Text, 3
1.12	Yes	Creation date (DateCreated)		Date
1.13	Yes	Tax entity (TaxEntity)	In a billing file it should be specified to which business establishment it applies. Otherwise it should have the value "Global”. In the integrated accounting file this field should have the value “Sede”	Text, 20
1.14	Yes	Fiscal identification of the entity that developed the software (ProductCompanyTaxID)	Should be the NIF of the company that developed the software.	Text, 20
1.15	Yes	Software certificate number (Software		Integer

		CertificateNumber)		
1.16	Yes	Application name (ProductID)	Name of the application that created the SAF-T file.	Text, 255
1.17	Yes	Application version (ProductVersion)	Version of the application that created the SAF-T file.	Text, 30
1.18	No	Additional comments (HeaderComment)		Text, 255
1.19	No	Telephone (Telephone)		Text, 20
1.20	No	Fax (Fax)		Text, 20
1.21	No	Email address (Email)		Text, 60
1.22	No	Company web address (Website)		Text, 60

Table 6 - SAF-T (Portuguese version) Header table (Ministério das Finanças, 2013).

Index	Required	Field name	Notes	Format
2.2.1	Yes	Customer identifier (CustomerID)	Customer unique identifier.	Text, 30
2.2.2	Yes	Account identifier (AccountID)		Text, 30
2.2.3	Yes	Customer tax identifier (CustomerTaxID)	Portuguese NIF.	Text, 20
2.2.4	Yes	Company name (CompanyName)	A generic customer can be identifier by “Consumidor final”.	Text, 100
2.2.5	No	Company contact name (Contact)		Text, 50
2.2.6	Yes	Billing address (BillingAddress)		N/A
2.2.6.1	No	Building number (BuildingNumber)		Text, 10
2.2.6.2	No	Street name (StreetName)		Text, 90
2.2.6.3	Yes	Detailed address (AddressDetail)	Should include the street name, building number and floor, if applicable.	Text, 100
2.2.6.4	Yes	City (City)		Text, 50
2.2.6.5	Yes	Postal code (PostalCode)		Text, 20
2.2.6.6	No	Region (Region)		Text, 50
2.2.6.7	Yes	Country (Country)	Should be the ISO 3166 1- alpha 2 country code.	Text, 12

2.2.7	No	Shipping address (ShipToAddress)		N/A
2.2.7.1	No	Building number (BuildingNumber)		Text, 10
2.2.7.2	No	Street name (StreetName)		Text, 90
2.2.7.3	Yes	Detailed address (AddressDetail)	Should include the street name, building number and floor, if applicable.	Text, 100
2.2.7.4	Yes	City (City)		Text, 50
2.2.7.5	Yes	Postal code (PostalCode)		Text, 20
2.2.7.6	No	Region (Region)		Text, 50
2.2.7.7	Yes	Country (Country)	Should be the ISO 3166 1- alpha 2 country code.	Text, 12
2.2.8	No	Telephone (Telephone)		Text, 20
2.2.9	No	Fax (Fax)		Text, 20
2.2.10	No	Email address (Email)		Text, 60
2.2.11	No	Website (Website)		Text, 60
2.2.12	Yes	Self-billing indicator	Should have the value “1” if there is a self-billing pact, otherwise “0”.	Integer

Table 7 - SAF-T (Portuguese version) Customers table (Ministério das Finanças, 2013).

4.2 Partial implementation of SAF-T (PT) report in OFBiz

OFBiz has the support for clean extensions, meaning that developers are capable of extending OFBiz with custom implementations without changing the standard files provided by the OFBiz developers. OFBiz has a folder called “hot-deploy” where developers should place the custom source-code. OFBiz allows users to extend almost all of the OFBiz components (entities, services, etc.) as well as change some of its default behaviour. In this case, to implement the SAF-T audit file generator we will create a whole new OFBiz module, the “Reports” module so we will not interfere with standard behaviours of other modules nor change the original OFBiz source-code. This allows OFBiz files to be updated without breaking our implementation (unless there are breaking changes on the framework).

All the source code for the implementation described below can be found in the following GitHub repository. The custom developments are contained in the “hot-deploy” folder.

```
https://github.com/JorgeAlm/ofbiz
```

This repository is a fork of OFBiz release 13.07 branch.

4.2.1 SAF-T (PT) Entities

As said above, the report that will be implemented won’t have all of the required information, we will only export the Header and the Customers tables. All of the required fields to generate the file are already present on OFBiz entities. To make the source-code simpler we will create new view entities with only the information that needs to be exported.

View name	Purpose
ReportSaftOrgInfo	Contains the party group name and tax information.
ReportSaftCustomers	Contains the customer’s name and tax information.
ReportSaftPartyPA	Contains the postal address information for all parties.
ReportSaftPartyTelecom	Contains the telecom information for all parties.
ReportSaftPartyWebContact	Contains the web contact information for all parties.

Table 8 – SAF-T (PT) model entities.

4.2.1.1 ReportSaftOrgInfo view entity

The ReportSaftOrgInfo view contains the group name and tax information for OFBiz party groups (party groups represent organizations). The Party that represents the current company should be a party group. If the OFBiz is created with the demo information the company party has the “Company” identifier. The following table contains the description of the fields that compose the ReportSaftOrgInfo view entity and what is the name of the entity where the field comes from.

Entity name	Field name	View field name	Comments
Party	partyId	partyId	Party identifier.
PartyGroup	groupName	groupName	Name of the party group.
PartyTaxAuthInfo	taxAuthGeoId	taxAuthGeoId	Geographical identifier of the tax authority.
PartyTaxAuthInfo	partyTaxId	partyTaxId	Party tax identification number.
PartyTaxAuthInfo	fromDate	taxInfoFromDate	Date after which the record is active.
PartyTaxAuthInfo	thruDate	taxInfoThruDate	Date after which the record becomes inactive. Null value

			indicates that the record is active forever.

Table 9 - ReportSaftOrgInfo view entity fields.

4.2.1.2 ReportSaftPartyPA view entity

The ReportSaftPartyPA view contains the postal address information for all existing parties. It is used to export postal address information for the company and for all of its customers. The following table contains the description of the fields that compose the ReportSaftPartyPA view entity and what is the name of the entity where the field comes from.

Entity name	Field name	View field name	Comments
Party	partyId	partyId	Party identifier.
PartyContact MechPurpose	contactMech PurposeTypeId	contactMechPurpose TypeId	The purpose type of the contact.
PostalAddress	address1	address1	The first line of the party postal address.
PostalAddress	address2	address2	The second line of the party postal address.
PostalAddress	city	city	The postal address city.
PostalAddress	postalCode	postalCode	The postal code.
PostalAddress	lastUpdated Stamp	paLastUpdatedStamp	The date when the Postal address record was updated.
Geo	geoId	stateGeoId	The geographical identifier of the state/region.
Geo	geoCode	stateGeoCode	The geographical code of the state/region.
Geo	geoName	stateGeoName	The name of state/region.
Geo	geoId	countryGeoId	The geographical identifier of the country.
Geo	geoCode	countryGeoCode	The geographical code of the country.
Geo	geoName	countryGeoName	The name of the country.
PartyContact Mech	fromDate	pcmFromDate	Date after which the PartyContact Mechanism record is active.
PartyContact Mech	thruDate	pcmThruDate	Date after which the Party contact mechanism record becomes inactive. Null value indicates that the record is active forever.
PartyContact MechPurpose	fromDate	pcmpFromDate	Date after which the Party Contact Mechanism purpose record is active.
PartyContact MechPurpose	thruDate	pcmpThruDate	Date after which the Party contact mechanism purpose record becomes inactive. Null

			value indicates that the record is active forever.

Table 10 - ReportSaftPartyPA view entity fields.

4.2.1.3 ReportSaftPartyTelecom view entity

The ReportSaftPartyTelecom view contains the telecom information for all parties. It is used to export telephone and fax numbers for the company and customers. The following table contains the description of the fields that compose the ReportSaftPartyTelecom view entity and what is the name of the entity where the field comes from.

Entity name	Field name	View field name	Comments
Party	partyId	partyId	Party identifier.
PartyContact MechPurpose	contactMech PurposeTypeId	contactMech PurposeTypeId	The purpose type of the contact.
TelecomNumber	countryCode	countryCode	The country code part of the telephone number.
TelecomNumber	areaCode	areaCode	The area code part of the telephone number.
TelecomNumber	contactNumber	contactNumber	The contact part of the telephone number.
TelecomNumber	lastUpdated Stamp	lastUpdated Stamp	Date when the Telecom number record was updated.
PartyContact Mech	fromDate	pcmFromDate	Date after which the PartyContact Mechanism record is active.
PartyContact Mech	thruDate	pcmThruDate	Date after which the Party contact mechanism record becomes inactive. Null value indicates that the record is active forever.
PartyContact MechPurpose	fromDate	pcmpFromDate	Date after which the Party Contact Mechanism purpose record is active.
PartyContact MechPurpose	thruDate	pcmpThruDate	Date after which the Party contact mechanism purpose record becomes inactive. Null value indicates that the record is active forever.

Table 11 - ReportSaftPartyTelecom view entity fields.

4.2.1.4 ReportSaftPartyWebContact view entity

The ReportSaftPartyWebContact view contains the web contact information for all parties. It is used to export the email and website addresses. The following table contains the description of the fields that compose the ReportSaftPartyWebContact view entity and what is the name of the entity where the field comes from.

Entity name	Field name	View field name	Comments
Party	partyId	partyId	Party identifier.
PartyContact MechPurpose	contactMech PurposeType Id	contactMech PurposeTypeId	The purpose type of the contact.
ContactMech	contactMech TypeId	contactMech TypeId	The identifier of the contact mechanism.
ContactMech	infoString	infoString	The contac information value.
ContactMech	lastUpdated Stamp	cmLastUpdated Stamp	Date when the contact mechanism was updated.
PartyContact Mech	fromDate	pcmFromDate	Date after which the PartyContact Mechanism record is active.
PartyContact Mech	thruDate	pcmThruDate	Date after which the Party contact mechanism record becomes inactive. Null value indicates that the record is active forever.
PartyContact MechPurpose	fromDate	pcmpFromDate	Date after which the Party Contact Mechanism purpose record is active.
PartyContact MechPurpose	thruDate	pcmpThruDate	Date after which the Party contact mechanism purpose record becomes inactive. Null value indicates that the record is active forever.

Table 12 - ReportSaftPartyWebContact view entity fields.

4.2.1.5 ReportSaftCustomers view entity

The ReportSaftCustomers view contains the customer's name and tax information. It is used to export the name of the customer and its tax information. This view filters all parties by the "CUSTOMER" role. The following table contains the description of the fields that compose the ReportSaftCustomers view entity and what is the name of the entity where the field comes from.

Entity name	Field name	View field name	Comments
Party	partyId	partyId	Party identifier.
Party	createdStamp	partyCreatedStamp	Date when the party record was created.
PartyRole	createdStamp	partyRoleCreatedStamp	Date when the party was associated with the role.
Person	firstName	firstName	Party first name.
Person	middleName	middleName	Party middle name.
Person	lastName	lastName	Party last name.

PartyGroup	groupName	groupName	Group name for the party group.
PartyGroup	groupNameLocal	groupNameLocal	Local group name for the party group.
PartyTaxAuthInfo	taxAuthGeoId	taxAuthGeoId	Geographical identifier of the party tax authority.
PartyTaxAuthInfo	partyTaxId	partyTaxId	Tax identification number.
PartyTaxAuthInfo	fromDate	taxInfoFromDate	Date after which the Party Tax Authority information purpose record is active.
PartyTaxAuthInfo	thruDate	taxInfoThruDate	Date after which the Party tax authority information record becomes inactive. Null value indicates that the record is active forever.

Table 13 - ReportSaftCustomers view entity fields.

4.2.2 Implementation architecture

Generating a report can be a long-running process, meaning that the user should have some feedback while the process is running. The OFBiz Job infrastructure is capable of running jobs synchronously, asynchronously or periodically but it does not return a reference to the job even if it is persisted. This means that we cannot schedule a job and receive its identifier, return a view to the user and update that view when the job finishes. The only way to achieve this is to specify a job name, schedule it to run only once and them later query the JobSandbox entity for that name to see if it is already finished and even then we could not have access to its result as the jobResult column of the JobSandbox entity is supposed to store simple string messages. In the case of the SAF-T audit file it would be useful to have validation messages returned to the user even if the audit file could not be generated, e.g. if some error is detected while exporting a given party the user should be notified about the error so it can be fixed (e.g. a badly formatted postal code). We cannot implement such requirements by relying solely on the current JobSandbox infrastructure so we are going to implement a custom infrastructure on top of that one.

4.2.2.1 Report Queue architecture

To overcome the limitations of the JobSandbox infrastructure we are going to implement a queue for reports that need to be generated. When the user intends to create a new SAF-T report we will create a new entry on the queue and trigger a Job associated with that queue entry. When the job finishes, it will persist the validation messages and the report result in custom entities. This removes the need to have a reference to the Job record while also leveraging the advantages of the current JobSandbox infrastructure (e.g. thread management and retry mechanism). Before going into the details of how this extension integrates with the OFBiz Job infrastructure we need to understand the entities behind it. The OFBiz job infrastructure is built on top of the JobSandbox entity. The following table shows the fields of the JobSandbox entity.

Field name	**Field type**	**Comments**
jobId	Identifier, not empty	Job identifier. Primary-key.
jobname	Text, 100 chars max	Job name.
runtime	Date-Time	Last execution date.
poolId	Text, 100 chars max	Thread pool identifier.
statusId	Text, 20 chars max	Job status identifier.
parentJobId	Text, 20 chars max	Parent job identifier.
previousJobId	Text, 20 chars max	Previous job identifier.
serviceName	Text, 100 chars max	Name of the service to run.
loaderName	Text, 100 chars max	Entity loader name.
maxRetry	Numeric	Maximum number of retries.
currentRetryCount	Numeric	Current number of retries.
runAsUser	Text, 255 chars max	User login to use when running the service.
runtimeDataId	Text, 20 chars max	Runtime data identifier. Foreign key to RuntimeData entity. Contains information such as locale and input other parameters.
recurrenceInfoId	Text, 20 chars max	Deprecated – should use tempExprId instead.
tempExprId	Text, 20 chars max	Information about the service recurrence. Foreign key to TemporalExpression entity.
currentRecurrenceCount	Numeric	Current recurrence count.
maxRecurrenceCount	Numeric	Max recurrence count.
runByInstanceId	Text, 20 chars max	OFBiz instance identifier where the job should be executed.
startDateTime	Date-time	Date when the job started running.
finishDateTime	Date-time	Date when the job finished.
cancelDateTime	Date-time	Date when the job was canceled.

jobResult	Text, 255 chars max	Result of the job.

Table 14 - JobSandbox entity fields.

To implement the requirements explained above while still integrating with the JobSandbox entities the following entities were created:

- ReportType – Contains all types of reports. Used to support more reports and not only the SAF-T that will be implemented.
- Report – Contains the content of the reports.
- ReportValidationMsgs – Contains the validation messages for the reports.
- ReportQueue – Queue of reports.

The following tables describe the entities enumerated above.

Field name	**Field type**	**Comments**
reportTypeId	Identifier, not empty	Report type identifier. Primary key.
description	Text, 255 chars max	Friendly name for the report type.
serviceName	Text, 255 chars max	The name of the service that is responsible for the generation of the report.
reportContentType	Text, 255 chars max	The HTTP Content type.
defaultFileName	Text, 60 chars max	The name that the file will have.

Table 15 - ReportType entity fields.

Field name	**Field type**	**Comments**
reportId	Identifier, not empty	Report identifier. Primary key.
reportTypeId	Identifier, not empty	Report type identifier.
reportName	Identifier, not empty	Name of the report.
createdByUserLogin	Identifier, not empty	User that created the report. Even though the report will be generated asynchronously by a job the user login that scheduled the job will be used.
reportData	Longest text field type supported by the database (e.g. CLOB in Oracle DBMS).	Report value in text format.
reportDataBin	Longest binary field type supported by the database (e.g. BLOB in Oracle DBMS).	Report value in binary format.

Table 16 - Report entity fields.

Field name	Field type	Comments
messageId	Identifier, not empty	Message identifier. Primary key.
reportId	Identifier, not empty	Report identifier.
messageSeverity	Text, 60 chars max	Message severity.
messageMapName	Text, 255 chars max	Identifier for the message in the labels map. Used to support internationalization of validation messages.
messageParameters	Longest text field type supported by the database (e.g. CLOB in Oracle DBMS).	Map<String, String> instance serialized as XML. Used as parameters for the message.

Table 17 - ReportValidationMsgs entity fields.

Field name	Field type	Comments
reportQueueId	Identifier, not empty	Report queue identifier. Primary key.
reportTypeId	Identifier, not empty	Report type identifier.
reportQueueStatusId	Text, 60 chars max	Report queue status identifier.
reportQueueParams	Longest text field type supported by the database (e.g. CLOB in Oracle DBMS).	Parameters of the report queue entry. Contains the parameters that will be sent to the service.
reportId	Text, 20 chars max.	Report identifier. Can be null if an error occurred during the report generation. In that case the report will exist, with validation messages and without data.

Table 18 - ReportQueue entity fields.

Now that the underlying entities were described we can take a look at how they interact with each other and with the OFBiz itself. To do so, the following figure illustrates the logic inside the "queueSaftPtReport" event that is responsible for the creating the report queue record and invoke the service responsible for processing report queue records, after user input.

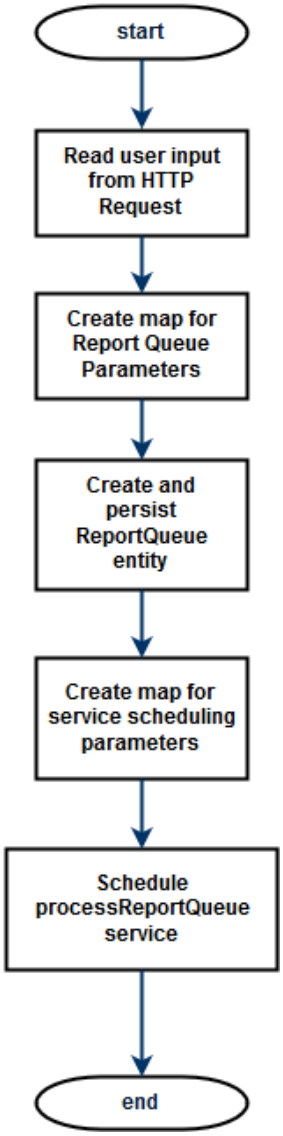


Figure 27 – Report Queue creation and processor service invocation logic.

After the ReportQueue is created the user is redirected to a new page, where the status of the report generation is displayed and periodically updated using AJAX. If the page is closed, the report generation is still running in the background. The “processReportQueue” service is responsible for deserializing the report queue parameters and invoke the service that actually generates the SAF-T audit file. This job will also update the report queue status, e.g. if the service that generates the SAF-T audit file fails unexpectedly with an unhandled exception this service will catch any exception and update the report queue entity accordingly. This service has only one input parameter, the report queue identifier since the result will be persisted on database tables.

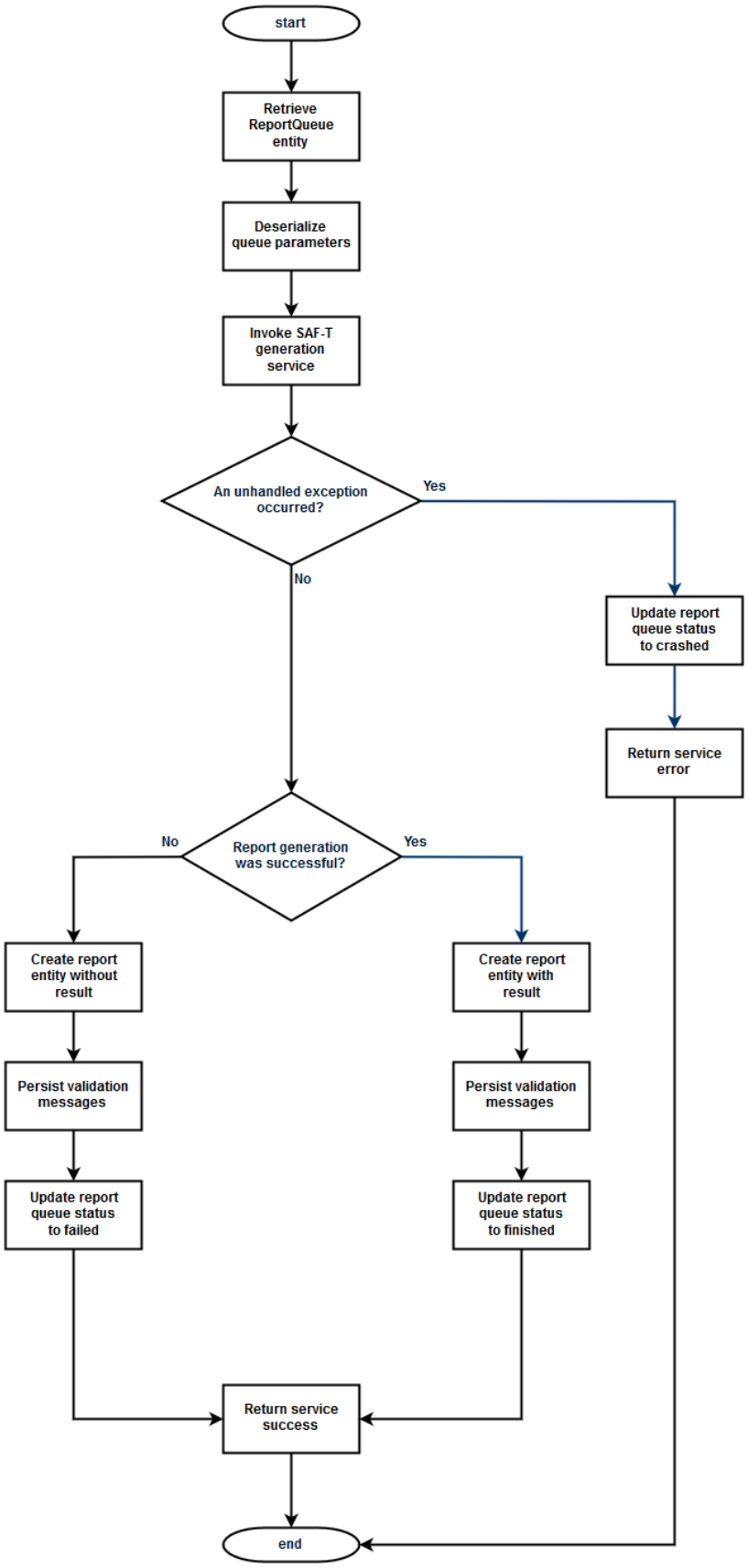


Figure 28 – Report queue record processor service logic.

The service responsible for generating the SAF-T audit file knows nothing about the underlying structure, meaning that if we want to run it using solely the JobSandbox infrastructure we still can. This service is named “generateSaft” and it is responsible for querying the database for the values, validating them and generate the XML content along the way. This service will return an error only if an unhandled exception occurs. All validation errors will be returned as validation messages.

The table below shows the details about the service definition attributes. All of the optional input parameters are there to allow users to override the default values used to gather information from the database. This is useful if the user creates specific contact mechanism purposes that are used to indicate what contacts should be used for reporting purposes.

Attribute name	Mode	Type	Optional?	Comments
timePeriod	Input	String	No	The identifier for the custom time period.
taxAuthGeoId	Input	String	Yes	Identifier for the Tax authority geographic location. Default value is “PRT”.
postalAddress PurposeTypeId	Input	String	Yes	Identifier for the purpose type of the postal address. Default value is “BILLING_LOCATION”.
phonePurpose TypeId	Input	String	Yes	Identifier for the purpose type of the phone number. Default value is “PRIMARY_PHONE”
faxPurposeType Id	Input	String	Yes	Identifier for the purpose type of the Fax number. Default value is “FAX_NUMBER”.
emailPurpose TypeId	Input	String	Yes	Identifier for the purpose type of the email contact. Default value is “PRIMARY_EMAIL”
websitePurpose TypeId	Input	String	Yes	Identifier for the purpose type of the website. Default value is “PRIMARY_WEB_URL”.
saftContent	Output	String	Yes	The XML string that represents the contents of the report file.
saftMessages	Output	Java.util.List	Yes	Instances of org.ofbiz.reports.ReportMessage. Contains the report validation messages.

Table 19 – generateSaft service attributes.

The following fluxogram illustrates the logic implemented in the SAF-T report generation service.

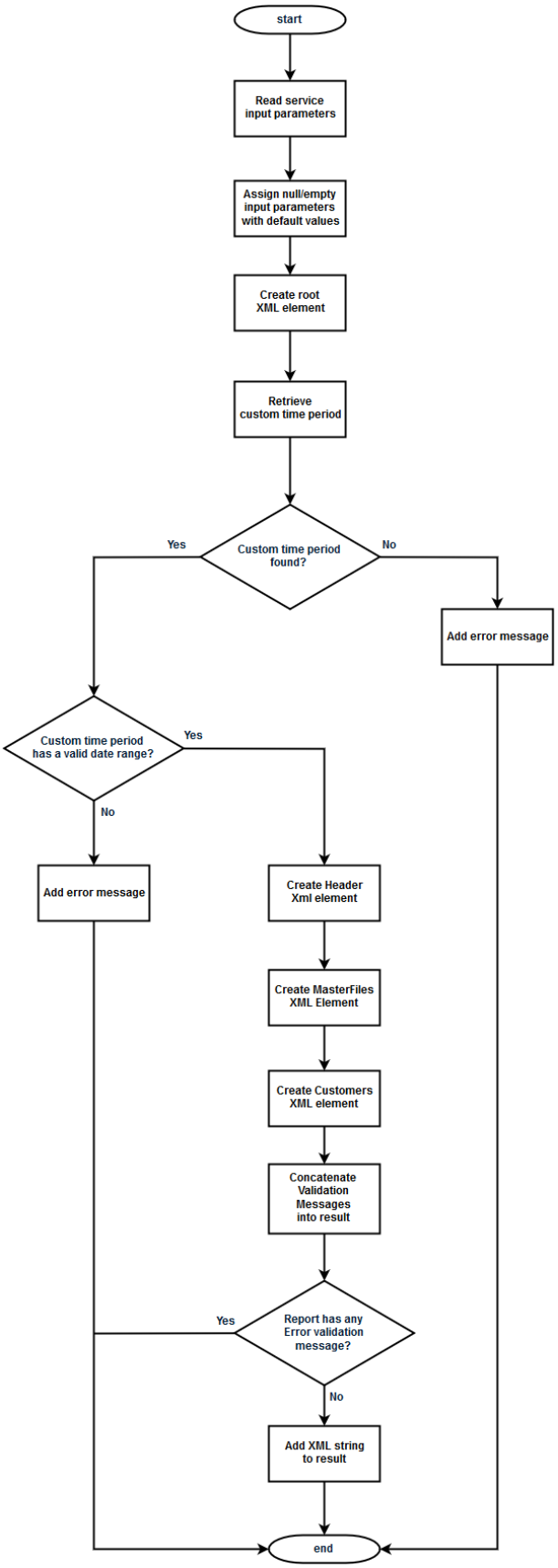


Figure 29 – generateSaft service logic.

This process can be slow when dealing with a huge number of records since it will fetch, for each party, the following in separate database queries:

- Party detail and tax information;
- Postal address information;
- Telecommunication contacts;
- Web contacts;

This represents four queries to the database by party. Ideally we should have only one, paginated query that would fetch all needed information from the database but with the current OFBiz framework that is not possible. Given that a party can have more than one postal address or contact mechanism we would need to have a view that would display only one postal address, the most recently created/updated one for example, for each party. To do this we would need to make use of a ROWNUMBER function with PARTITION and ORDER BY. OFBiz does not support this type of functions because the underlying DBMS that it can be used with don't support such functions (e.g. the default Derby DBMS[11]).

4.2.3 Reports module user interfaces

The reports module, as any other OFBiz module can be accessed using the main menu, as shown below.

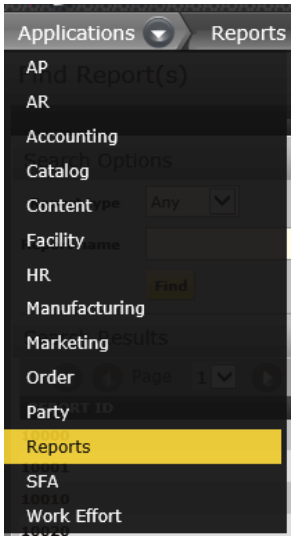


Figure 30 – OFBiz main menu integration.

After clicking the Reports menu, the sub-menu will display three options:

- Main
- Find Reports(s)
- SAFT-PT

[11] http://db.apache.org/derby/docs/10.8/ref/rreffuncrownumber.html

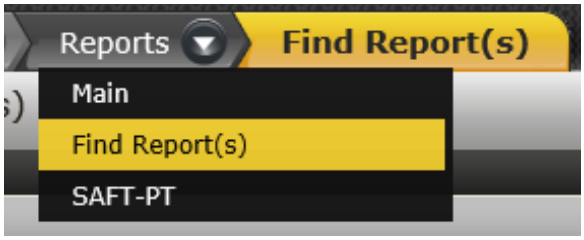


Figure 31 - OFBIz Reports sub-menu.

4.2.3.1 Main/Find Report(s) screen

Clicking the “Main” or the “Find Report(s)” option will display the same page. This page will show a list of the report that the current user has generated in the past and allows users to search that list. The supported filters are the report type and report name, even though at this point only one report type is available. This screen makes use of common UI components, such as the pagination display and follows the OFBiz user experience guides. The table contains an actions column that allows users to quickly download or delete the report as well as go to the report details page where the validation messages among other information about the report are displayed.

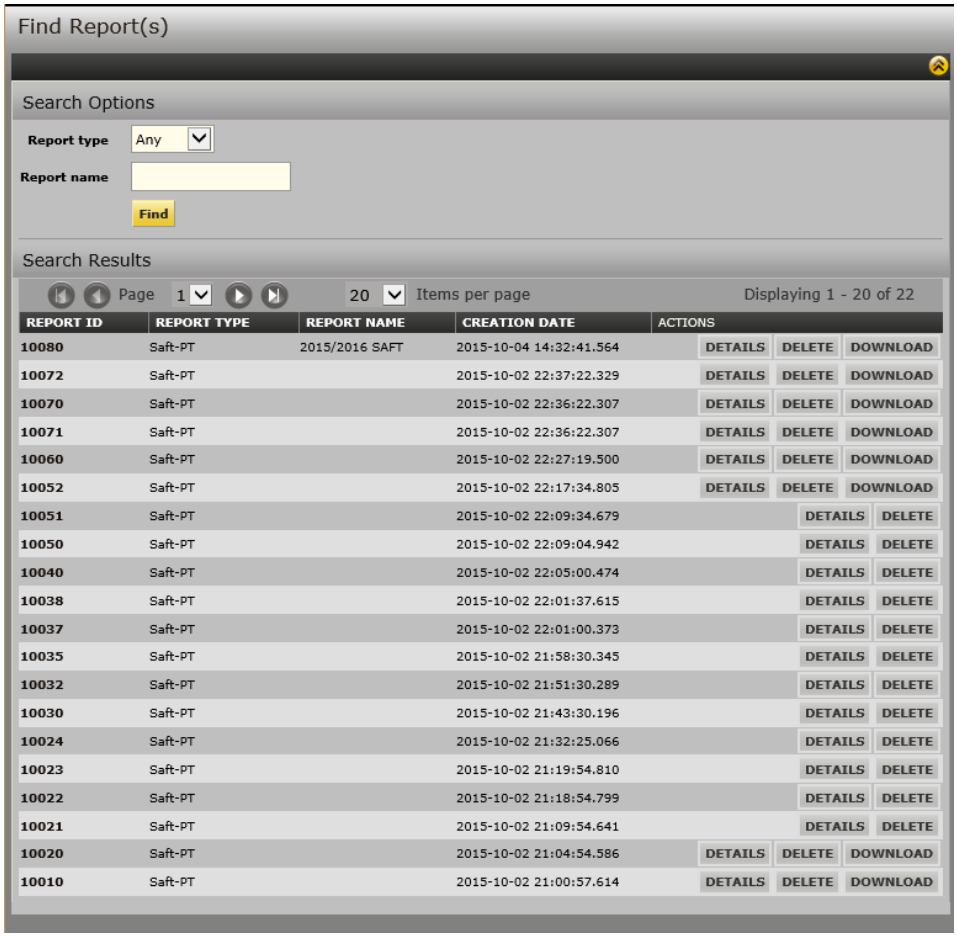


Figure 32 – Find Report(s) screen.

4.2.3.2 SAFT-PT screens

The SAFT-PT main screen is where the user is capable of creating new SAF-T PT reports. This screen displays only a simple form with basic information needed to generate the report. The form contains the following fields:

Figure 33 – SAFT-PT Report creation screen.

Of the fields displayed on the image above, only the Fiscal Year is required. The report name is a free text input field for users to specify a friendly name for the report. All other fields are used to override default parameters of the report generation service as it was explained in the chapters above. Clicking the submit button will trigger the "queueSaftPtReport" event that will create the report queue record and schedule the "processReportQueue" service to run as soon as possible. If everything is successful, the user will be redirected to the report details page that will show a spinning circle until the report is generated or fails.

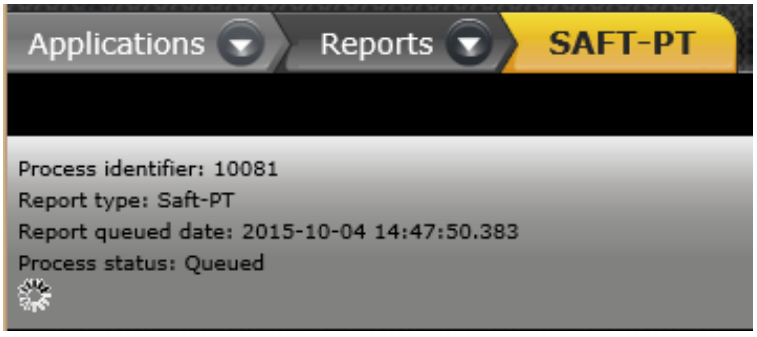


Figure 34 – Report details page with Queued report.

This page will fetch the process status via AJAX every 5 seconds. This page exists to give users some feedback if the report generation takes a long time, which can happen when exporting lots of records. Once the report service is finished (either successfully or not) another AJAX request will be made to retrieve the report validation messages and the download/delete

buttons. This screen is the same that will be displayed if the user clicks on the “Details” button on the table present on the Find Report(s) page.

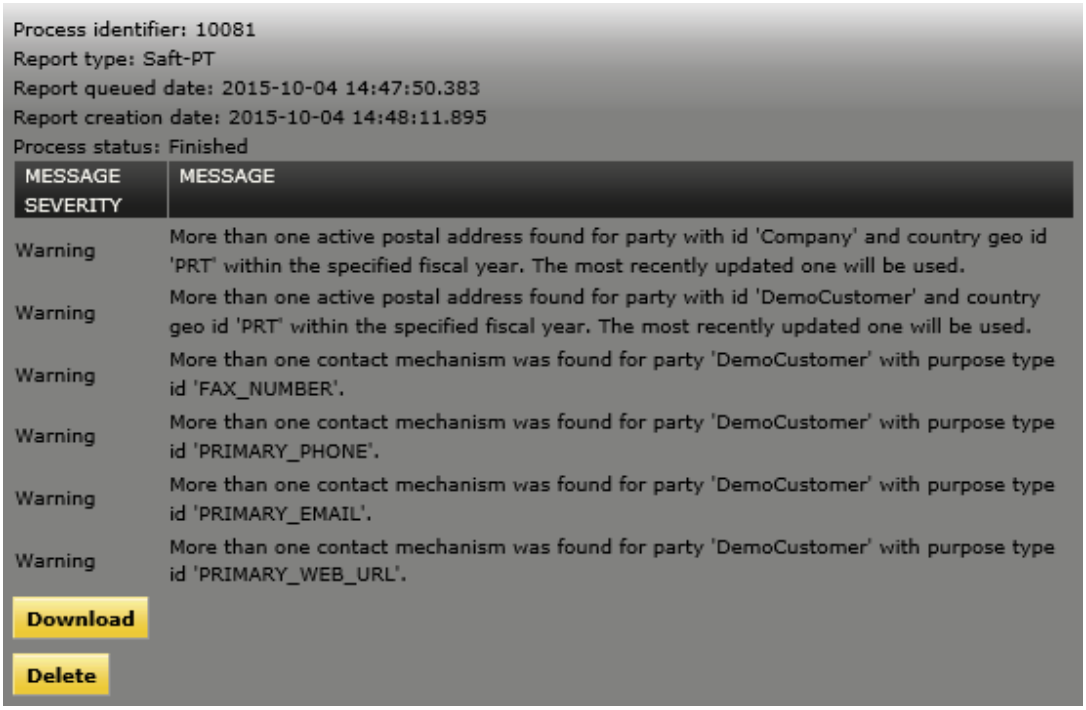


Figure 35 – Report details screen.

4.2.3.3 Integration with OFBiz

This chapter will describe how the module was integrated with the OFBiz UI by using the common screen decorator and with the authentication methods by using common action pre-processors.

Screen decorators

OFBiz already provide a decorator that is used by every module but it is not ready to use, it has some specifications that vary by module, e.g. the application titles and menus.

The following XML shows the basic decorator that needed to be declared and that makes use of the decorator that is provided by OFBiz.

```
<screen name="main-decorator">
  <section>
    <actions>
      <property-map resource="CommonUiLabels"
            map-name="uiLabelMap" global="true" />
      <property-map resource="ReportsUiLabels"
            map-name="uiLabelMap" global="true" />
      <set field="layoutSettings.companyName"
            from-field="uiLabelMap.ReportsCompanyName"
            global="true" />
      <set field="layoutSettings.companySubtitle"
            from-field="uiLabelMap.ReportsCompanySubtitle"
            global="true" />
      <set field="applicationMenuName"
            value="ReportsAppBar" global="true" />
      <set field="applicationMenuLocation"
            value="component://reports/widget/reports
                  /ReportsMenus.xml"
            global="true" />
      <set field="applicationTitle"
        value="${uiLabelMap.ReportsApplication}"
        global="true" />
    </actions>
    <widgets>
      <include-screen name="ApplicationDecorator"
        location="component://commonext/widget
            /CommonScreens.xml" />
    </widgets>
  </section>
</screen>
```

A simple screen using this decorator could be declared as follows.

```
<screen name="ReportSaft">
  <section>
    <widgets>
      <decorator-screen name="main-decorator"
        location="${parameters.mainDecoratorLocation}">
        <decorator-section name="body">
          <label text="Content…"></label>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

Menus

OFBiz back-end pages display a menu on the top that changes depending on the page that is currently being displayed. These menus must be defined in an XML file named "<module>Menus.xml" that is placed in the same folder as the module screens XML file.

Menus are declared using the root menus element and each menu is declared using the menu element. OFBiz menus should extend the “CommonAppBarMenu” menu and the “component://common/widget/CommonMenus.xml” resource.

Menu items can be composed of other menus or hyperlinks, using the link element as shown below:

```
<menus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/
      widget-menu.xsd">
  <menu name="ReportsAppBar" title="${uiLabelMap.Reports}"
    extends="CommonAppBarMenu"
    extends-resource="component://common/widget/CommonMenus.xml">
    <menu-item name="findReports"
        title="${uiLabelMap.FindReportsMenuTitle}">
      <link target="FindReports"/>
    </menu-item>
    <menu-item name="reportsaft" title="${uiLabelMap.ReportSaft}">
      <link target="ReportSaft" />
    </menu-item>
  </menu>
</menus>
```

This menu is then displayed on the page by setting the applicationMenuName and applicationMenuLocation fields of the common ofbiz decorator as shown in the simple screen decorator above.

Authentication

If we are developing a new module for the OFBiz backoffice we need to make sure that the user is logged in and if not, redirect the request to the OFBiz login page. The controller specification has two main tags:

- Preprocessor – logic that must be executed before any request to the module;
- Postprocessor – logic that must be executed after any request to the module.

To allow access to the module only to authenticated users, a new event must be fired before all requests. This event is a common OFBiz event that is used in all application modules of the OFBiz back-office and it would be registered as follows:

```
<preprocessor>
  <!-- Events to run on every request before security
       (chains exempt) -->
  <event name="check509CertLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="check509CertLogin" />
  <event name="checkRequestHeaderLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkRequestHeaderLogin" />
  <event name="checkServletRequestRemoteUserLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkServletRequestRemoteUserLogin" />
  <event name="checkExternalLoginKey" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkExternalLoginKey" />
  <event name="checkProtectedView" type="java"
    path="org.ofbiz.webapp.control.ProtectViewWorker"
    invoke="checkProtectedView" />
  <event name="extensionConnectLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="extensionConnectLogin" />
</preprocessor>
```

These preprocessor events are required for all controllers and as such are defined in the /framework/common/webcommon/WEB-INF/common-controller.xml file. To include this behavior in the new module, the controller.xml file must contain the following line at the beginning (before the description element).

```
<include location="component://common/webcommon/WEB-INF/common-
controller.xml" />
<description>Reports Module Site Configuration File</description>
```

When changing from one module to another, an external login key must be specified in the URL parameters for the user to stay logged in.

```
https://127.0.0.1:8443/humanres/control/main?externalLoginKey=EL460517773307
```

The above preprocessor is intended to be used to share a login key between application modules but it does not automatically restrict access to authenticated users.

To allow access only to authenticated users to any given request, the “auth” attribute of the “security” element must be specified in the request definition with a value of “true”.

The “auth” attribute works by firing a request to the “checkLogin” action of the common OFBiz control servlet. This action will validate if the user is logged in and if not, the user will be redirected to the login page.

When dealing with user credentials and other sensitive data in web applications the HTTPS protocol should be used to ensure proper security over the communication between the end-user and the server. OFBiz can work with both HTTP and HTTPS, which is opt-in. To require a

secure connection to fire a given request the “https” attribute of the “security” element must be specified in the request-map definition, as shown below:

```
<request-map uri="viewprofile">
  <security https="true" auth="true" />
  <response name="success" type="view" value="viewprofile"
    save-home-view="true" />
</request-map>
```

4.2.3.4 Internationalization support

All of the screens displayed above use the OFBiz internationalization support and can be displayed in English as well as in Portuguese. To change the language that the screen is displayed in you must change the language preferences for your current session:

Figure 36 – OFBiz language preferences menu entry.

On the presented screen choose, Portuguese (pt) and all of the screens will now be displayed in Portuguese.

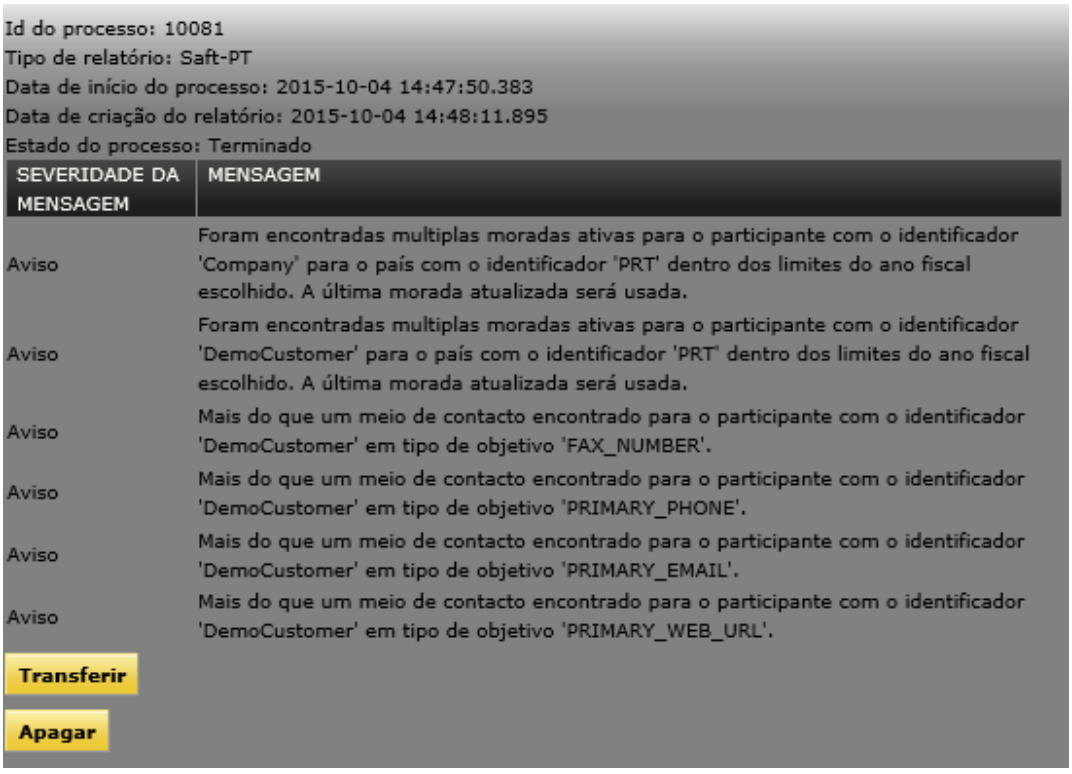


Figure 37 - Report details screen (portuguese).

Figure 38 - Find Report(s) screen (portuguese).

Figure 39 - Report creation screen (portuguese).

4.3 Tests

At this point, and because we are dealing with a prototype application, no unit tests were developed yet. All the tests made to the current implementation were manual tests relying on the SAF-T (PT) audit file validation tool[12] provided by the Portuguese government.

In the future we should develop unit tests that simulate specific scenarios. On top of the unit tests we could also develop some WebDriver based tests (Browser Automatition tests) using, for example Selenium[13], to test the user interface and integration between pages.

The Appendix II details the manual tests that were made to the current implementation.

[12] http://info.portaldasfinancas.gov.pt/apps/saft-pt03/
[13] http://www.seleniumhq.org/

4.4 Summary

The first topic gives a detailed description of the Portuguese version of the SAF-T audit file, showing what information it should contain and how the contents can vary depending on the purpose of the system that creates the audit file. We detailed only the Header and Customers tables as these are the only tables that we exported from OFBIz.

The second topic shows the components that were developed to support the generation of SAF-T audit files from OFBiz. This includes creating new model entities that are used to overcome limitations of the current implementation of the background job infrastructure on OFBiz as well as view entities that are simply used to gather only the information that the audit file generation service will export and use to validate the contents that are being exported. This topic contains only the documentation of the different entities, web pages and services that were created, the source code is available at the GitHub repository.

The last topic details how the implemented solution was tested to guarantee its quality. Since the solution covers only a part of the SAF-T audit file only manual tests we're made. In the future we should develop unit tests and browser automation tests.

5 Analysis and results

Now that we fully understand how to adapt OFBiz with new features we can specify what would need to be developed in order to have a party management system capable of being used in Portugal, on a web-based environment using Java as the programming language without using OFBiz. In this chapter we will describe what would take to implement a simple Java Party management application that respects the Portuguese law in contrast to adapting OFBiz to respect the same laws.

As we have seen above, OFBiz provides a more specific framework with a higher level of functionality when compared to Java.

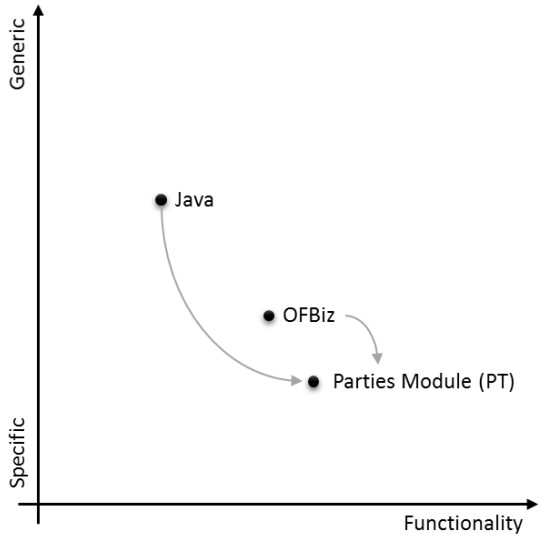


Figure 40 - Framework comparison chart.

This means that when opting for developing an application using Java instead of OFBiz we are going to have to spend time thinking about the application architecture and implementing core features such as database creation and integration.

For this comparison we're going to assume that the developer knows nothing about the OFBiz framework and is familiar with Java programming language and all of the frameworks referenced below. Also, the Java application would contain only the features of the Parties module and some other core features of other modules (e.g. accounting).

To create this fit-to-purpose website the developer would have to create the Data Model, either by using the data models referenced in The Data Model Resource Book (Silverston, 2001) or by creating custom data models. The database would need entities to support at least party management and some of the accounting module. This means that on the website we would also need to implement screens to maintain all of this information. To help with the persistence we could use the Hibernate framework[14] and for the website we could use the Java Spring MVC[15] framework. This would reduce some of the plumbing code but there would still be a lot of development time before starting the implementation of the Parties module. The table below shows the tasks that would need to be executed in order to have a minimalistic application capable of dealing with the requirements of the Portuguese law.

Task	Estimated time (hours)
Application UI concepts	40
Data model creation	40
Java integration with the data model	40
Security framework	40
Service engine	40
Core MVC structure (incl. base pages)	80
Data maintenance pages	200
Portuguese law requirements	200
Total	680 Hours / 85 Days – 8hrs work per day

Figure 41 – Custom application estimated time.

OFBiz provides a good development framework and a consistent user experience, anyone trying to develop a custom application would need to spend some time taking care of the user interface and user experience and even if front-end frameworks such as Bootstrap[16] are used, some work would still need to be done in order to provide a coherent user experience. Approximately one week would be spent choosing the most appropriate front-end framework, deciding whether or not to support mobile devices, choosing or developing a theme and doing some mock-ups of different types of screens (e.g. a screen used to search a given entity and a screen used to change the values of an entity).

[14] http://hibernate.org/
[15] https://spring.io/
[16] http://getbootstrap.com/

The second week would be dedicated to the Data model. This implies that the user must get familiar with the data model used in The Data Model Resource Book (Silverston, 2001) or will create a custom data model from scratch. In this task all database entities would need to be created if the framework for the persistence layer of the application is not capable of creating database entities on its own. In this case the estimated time assumes that the user would create the data models of the book and that the persistence layer framework is capable of mapping Java objects to database objects. Most of the time, if not all would be spent understanding and adapting the data models to respect the Portuguese law.

The third week would be used to create the Java domain model classes for all database objects. Again, the estimated time assumes that the persistence layer framework is capable of creating database objects on its own and provides a Java API for querying those objects, removing the need to develop stored procedures.

After having a domain model created we need to choose how the security of our application will work. OFBiz already provides a security framework that can be based on roles, users, and single permissions giving great control to the application administrators and an easy way to control who can view some information. Here we would need to decide whether or not we should implement a robust security framework or implement a simpler one, e.g. implement security simply based on roles. In this task we also need to design the security data model, e.g. Users table, user roles, etc.

Other important feature that our application would need is the capability of running recurrent jobs on the background. The estimated time for this task contemplates the time to develop the data model supporting the background job engine as well as the Java API used to schedule and maintain background jobs.

Now that all of our core features are implemented we can start building the website, for example using the Java Spring MVC framework. This framework implements a Model-View-Controller pattern in Java, is simple to use and flexible. At this point we should implement the base pages, or page fragments, for the website. These page fragments represent portions of page that will be reused throughout the application, e.g. the header, menus, templates for displaying action feedback to user, etc.

The next step would be to implement the maintenance screens for all of the entities that represent the data model and that need to be maintained by the application users. This represents screens such as the Party search screen, Party editing screen, etc.

Until this point we were only implementing features similar to the ones that OFBiz already provide out-of-the-box in the Parties module but our development would already be respecting the Portuguese law and it might not support other countries. Now would be the time to implement custom features, such as the SAF-T PT audit file generator, that OFBiz doesn't have.

If we use the OFBiz framework, the development time will be smaller but we have to take into account the time that would take to learn how everything works in OFBiz before starting to

develop anything. The table below shows an estimation of the time that would take to fully adapt OFBiz to respect the Portuguese law.

Task	Estimated time (hours)
OFBiz study (Incl. data model)	120
Data model extensions/adaptations	24
Standard screen/code adaptations	40
Portuguese law requirements	320
Total	504 Hours / 63 Days – 8hrs work per day

Figure 42 - OFBiz adaptation estimated time.

First, we would need to study the Data model that OFBiz uses and then we could start looking at the OFBiz source-code which would involve some early developments to find out how some things works as well as looking at the source-code to see how something is implemented.

After fully comprehending the OFBiz framework we could start thinking about what entities would need to be changed in order to support our requirements. Along the way we would also end-up changing some screens for example to show more fields or provide other required features. The data model and standard screen adaptions would probably end-up being merged.

Lastly we could implement features that OFBiz doesn't contemplate. This tasks would be used to develop maintenance screens for new entities that OFBiz doesn't have, new reports, etc. When compared to the development using Java the time needed for this task is bigger since the Java developments know what entities it will need from the beginning and can plan what maintenance screens will be implemented while in OFBiz we can adapt the screens that already exist but we will need to create some new screens because OFBiz might not even know certain entities.

Now that we know the fundamentals of ERPs, how OFBiz works and how it can be extended as well as how we could develop a simple Party module from scratch we can start answering the questions that led to this investigation.

One of the first question that arose was related to the number of features provided by an Open-Source ERP when compared to a commercial one. Based on the details provided above from OFBiz and from the SAP ERP we can conclude that the SAP ERP provides a greater deal of features but not all of them are required by small or medium companies, for those OFBiz or other open-source might still be a viable option. This varies from company to company and each case should spend time investigating what ERP better suits their needs.

The main question asked if it would be better to create an application from scratch or to adapt an existing open-source one. Based on the prototype implementation of the SAF-T PT audit file in OFBiz we saw that it is not difficult to extend OFBiz but it takes some time to learn the existing framework. The answer to the question is actually whether or not it is quicker to learn how the open-source solution works and if/how it can be extended than building a similar framework from scratch. From the information we gathered above we can see that we have less tasks to do if we opt for adapting an existing solution and even if the time to learn the open-source

solution is higher than starting from nothing it would be wiser to choose the open-source solution for two reasons:

1. It already contains a greater deal of features;
2. It has a community improving and support it;

This means that as the number of people using and supporting the software grows, the features will grow with it and you will benefit from it without having to write a single line of code.

5.1 Summary

This chapter summarises the capabilities that are provided by OFBiz and the efforts that were needed to extend the application to support custom requirements. Based on the features that OFBiz contains and what is expected of an ERP we enumerate the tasks that would need to be executed in order to develop a custom web application capable of managing parties. The estimated times assume that developers would be familiar with the Java programming language as well as with the frameworks indicated above. We then compare the estimated amount of time that would be needed to develop a custom solution and compare it with an estimation of the time that a developer without any knowledge of the OFBiz ERP would need to extend OFBiz to support the same requirements.

The contents of the current chapter provide an answer to the first research question (RQ1) by providing estimated times for both the development of a custom application from scratch as well as to extend OFBiz. We saw that to develop a custom application we would need to implement many of the features that OFBiz already provides such as a security module, a background job infrastructure, etc. The time that would be spent developing these components would exceed the time needed to learn the OFBiz framework.

6 Conclusion

This chapter will conclude the analysis demonstrated above by summarizing some key facts that were highlighted in the previous chapters.

First, by analysing the generic data models and specifications for Enterprise Resource Planning software we can see that these types of applications are highly complex and have a broad scope, implementing processes from distinct functional areas of a company. By automating business processes, it is also possible to enforce business rules. The key advantage that an ERP provides is the fact that it has a central database and presents the right information to right people at the right time, facilitating the decision making process.

When dealing with ERPs it's important to find the right one for our needs. In the chapters above we compared the features provided by an open-source solution and compared it to a widely used commercial one. We reached the conclusion that the commercial one provides a lot more features. Taking into account that commercial ERPs are usually not affordable by small to medium companies we wanted to see how viable it would be to use an open-source ERP. To do that we investigated the features of the Parties module and concluded that it does not provide all the features that a Portuguese company needs. This led to the development of a prototype module capable of generating the SAF-T PT audit file. By doing this we can have an idea of the effort that would be needed to adapt and use the OFBiz in Portugal. Given that OFBiz has a lot of modules and already has its own framework and domain specific languages developers would need to spend time learning these frameworks before they are able to change the application. So, in the end we proposed another approach which would be developing the Parties module from scratch using Java and other frameworks such as Hibernate and Spring MVC, which proved to be more time consuming than adaping OFBiz.

6.1 Research evaluation

This chapter will reflect upon the research questions and provide answers to them.

RQ1 - *What would be the effort, if any, to adapt an open-source ERP to local law compared to developing a fit-to-purpose application from scratch?*

Based on the research contained in chapter 5, we can conclude that the effort of implementing a fit-to-purpose application can be higher than adapting an open-source ERP. The facts presented show that developing a custom application capable of managing Parties would be higher than learning OFBiz and extending it with new requirements. Implementing a custom solution would require a lot of development time to implement features and components that OFBiz provides without any kind of extension.

RQ2 - *How does an open-source ERP compare to a commercial one in terms of features?*

The analysis of the SAP ERP and Apache OFBiz in the second and third chapters provide enough information to conclude that SAP ERP provides more features than OFBiz. These chapters describe the application modules that compose each of the ERPs, and we can see that even though they seem to cover the same functional areas, the modules of the SAP ERP are more complete.

LQ1 - *What is an ERP and what features should it provide?*

The answer to this question can be found in the second and third chapters. The analysis of the functional areas that should be covered by an ERP gave a brief idea of the features that an ERP should have. This was complemented by the research of the main features that are supported by each of the application modules of SAP ERP and Apache OFBiz.

LQ2 - *How does the data model of an open-source ERP compare to the one of a commercial ERP?*

This question was answered in the second chapter. We can conclude that the data model of SAP ERP is a lot different from the one used by Apache OFBiz. SAP ERP does not have the concept of Party, instead each party type (e.g. employee or customer) has its own entity, stored in distinct database tables while OFBiz follows the generic data model presented in the same chapter.

6.2 Summary

The comparison between a commercial and open-source ERP provided an insight of the features and modules of each one and we reached the conclusion that the commercial ERP provides more features and covers more business requirements.

When it comes to developing a fit-to-purpose application, we concluded that it would take more time to develop a whole new framework than it takes to learn how OFBiz works but this can vary from case to case, depending on the requirements of the final product and technologies

used. Every case should be analysed independently but extending an existing framework proved to be less time consuming and offer a greater number of features. This documents aims to provide a guide to make the decision between a commercial ERP, an open-source and the development of a custom application and as such every scenario should be analysed on its own as the requirements for the final product may vary.

6.3 Future Work

Even though the partial implementation of the generator of SAF-T audit files was enough to prove that OFBiz is easely extensible and that developing a custom solution from scratch would be more time consuming than adapting OFBiz, we should investigate other requirements of the Portuguese law that OFBiz might not contemplate. Also, the SAF-T audit file generator should be completed to include all of the information that the SAF-T file supports, at least for a single audit file purpose. This would probably lead to other extensions such as adding new fields to some existing entity and application screen. It would also be usefull if this type of analysis was made to the other application modules.

The technical documentation should be made available to the OFBiz community as it might help improving the adoption of the software. The guide should be extended to describe some “minor” components such as the OFBiz themes.

7 References

Beginners-SAP. (28 de May de 2015). *What does 4 in ABAP 4 stand for ?* Retrieved from Beginners-SAP: http://www.beginners-sap.com/what-does-4-in-abap-4-stand-for/

Christian Krämer, S. R. (2005). *Mastering HR Management with SAP: The complete guide to SAP HR.* SAP PRESS.

Cleven, A., Gubler, P., & Hüner, K. (8 de May de 2009). *Design Alternatives for the Evaluation of Design Science Research Artifacts*. Retrieved from ResearchGate: http://www.researchgate.net/publication/221581374_Design_alternatives_for_the_evaluation_of_design_science_research_artifacts

Columbus, L. (12 de May de 2014). *Gartner's ERP Market Share Update Shows The Future Of Cloud ERP Is Now*. Retrieved from Forbes: http://www.forbes.com/sites/louiscolumbus/2014/05/12/gartners-erp-market-share-update-shows-the-future-of-cloud-erp-is-now/

Ellen Monk, B. W. (2012). *Concepts in Enterprise Resource Planning.* Cengage Learning.

Hoffman, R. (2010). *Apache OFBiz cookbook.* Packt Publishing.

Jonathan Wong, R. H. (2008). *Apache OFBiz Development.* Packt Publishing.

Jones, D. (10 de May de 2015). *Apache OFBiz™ Project Overview*. Retrieved from Apache OFBiz™: https://ofbiz.apache.org/apache-ofbiz-project-overview.html

Ministério das Finanças. (21 de Ago de 2013). Portaria n.º 274/2013.

SAP-ERP. (s.d.). *SAP Module overview*. Retrieved from SAP-ERP: http://www.sap-erp.com/general/sap-module-overview.html

Silverston, L. (2001). *The Data Model Resource Book Revised Edition Volume 1.* Wiley Computer Publishing.

The Apache Software Foundation. (08 de Oct de 2015). *Apache OFBiz™*. Retrieved from https://ofbiz.apache.org/

The Apache Software Foundation. (08 de Oct de 2015). *OFBiz Project Open Wiki*. Retrieved from https://cwiki.apache.org/confluence/display/OFBIZ/Home

The Apache Software Foundation. (08 de Oct de 2015). *OFBiz Technical Documentation*. Retrieved from https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Technical+Documentation+-+Home+Page

Vaishnavi, V., & Kuechler, B. (23 de Oct de 2013). *Design Science Research in Information Systems.* Retrieved from Association for Information Systems: http://desrist.org/design-research-in-information-systems/

Appendix I

Instituto Superior de Engenharia do Porto

Apache OFBiz™ Technical Documentation

Jorge Almeida
25-10-2015

Table of Contents

1 Introduction

OFBiz is an open-source ERP implemented in Java. The application makes use of a lot of existing third-party components and frameworks, like Apache Tomcat and JDBC. The following chapters will describe the core application components, explain how they interact with each other and showing source-code examples. All the contents of this document are based on Apache OFBiz version 13.07.

2 OFBiz eclipse project

The eclipse project for the OFBiz is composed of the following folders:

- applications – this folder contains the definition of the entities, services and user interface for each of the OFBiz application modules. This is where the core business logic is located.
- framework – this folder contains all the source code of the OFBiz framework.
- hot-deploy – this folder contains components that will be loaded after the OFBiz components (custom application modules should be created here)
- lib – contains the libraries used by OFBiz.
- runtime – this folder contains the logs and output files.
- specialpurpose – this folder contains special purpose built modules, like the e-commerce module.
- themes – this folder contains the user interface themes that are common to all OFBiz application modules (like the main application bar, footer, cascading style sheets, etc.)
- tools – this folder contains command-line scripts used to start/stop ofbiz as well as manage SVN/Git folders.

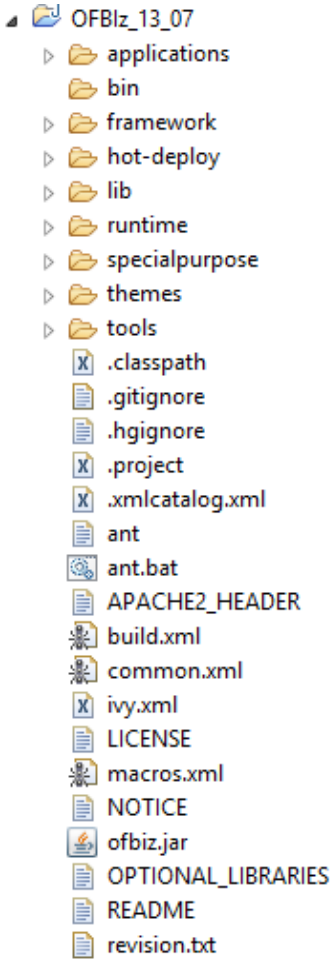


Figure 1– OFBiz project structure

2.1 Application module structure

Inside the applications folder there are sub-folders for each of the OFBiz modules. Each module can contain the following folders:

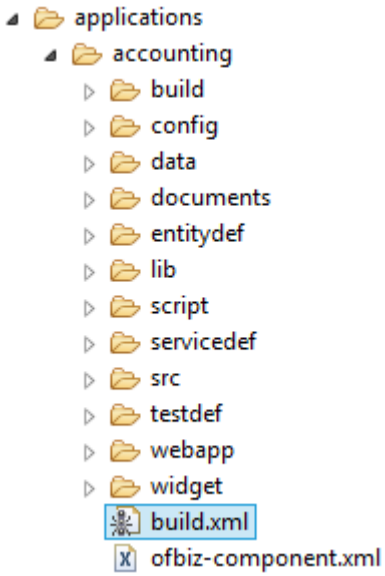


Figure 2 – OFBiz application module structure

- build – contains the result of the compilation of the application module source-code.
- config – this folder contains configuration files used within the module, e.g. application texts.
- data –contains seed, or test data files as well as help/documentation files.
- documents – contains the xml definition for help documents.
- entitydef – contains the entity definition files.
- lib – contains external libraries.
- script – contains mini-language scripts.
- servicedef – contains the service definitions.
- src – contains all java source-code files.
- testdef – contains test scenarios.
- webapp – contains the web application itself. It contains the definition of actions and views.
- widget – contains the definition of the application views.

The ofbiz-component.xml file is used to let OFBiz know what is contained within the module and where it is located.

The build.xml file is used to instruct ANT how the application module should be compiled.

2.2 Compiling OFBiz

This chapter will show the required steps to be able to change and compile OFBiz source code. We will be using Eclipse Luna as our development environment.

The first thing to do is display the “Navigator” view. This view is the ideal view to work in OFBiz as it displays the folder structure as it is in the workspace and it is much cleaner than the package view. To do this go to the “Window” menu > Show View and select the “Navigator” option.

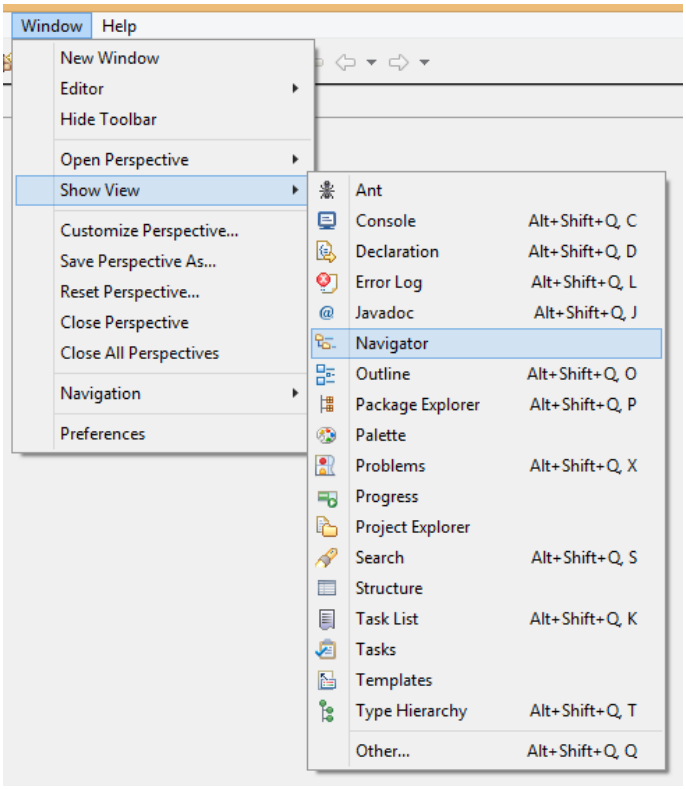


Figure 3 – Showing Eclipse Navigator view

The next step is to import the actual source-code. This can be done by right clicking inside the navigator view and clicking “Import”.

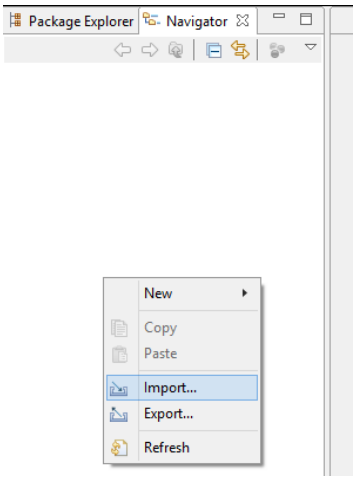


Figure 4 – Importing OFBiz source code to Eclipse

A new window should appear prompting to select the import source. Select “Existing projects into workspace”

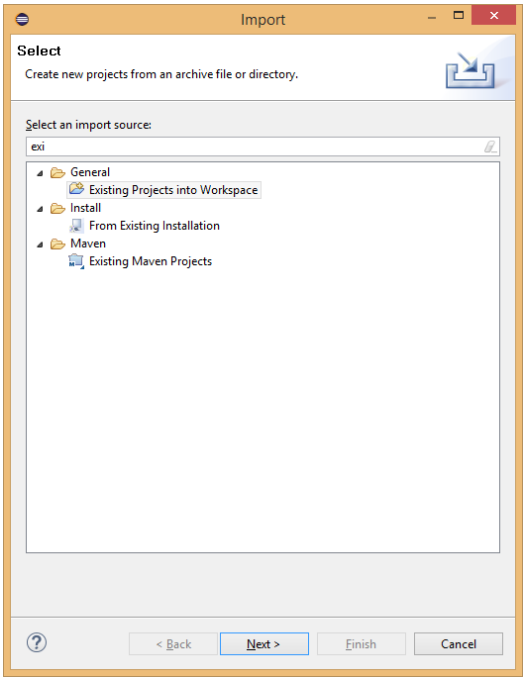


Figure 5 – Selecting the Eclipse import source

Click “Next” and select the folder where the OFBiz source-code is located.

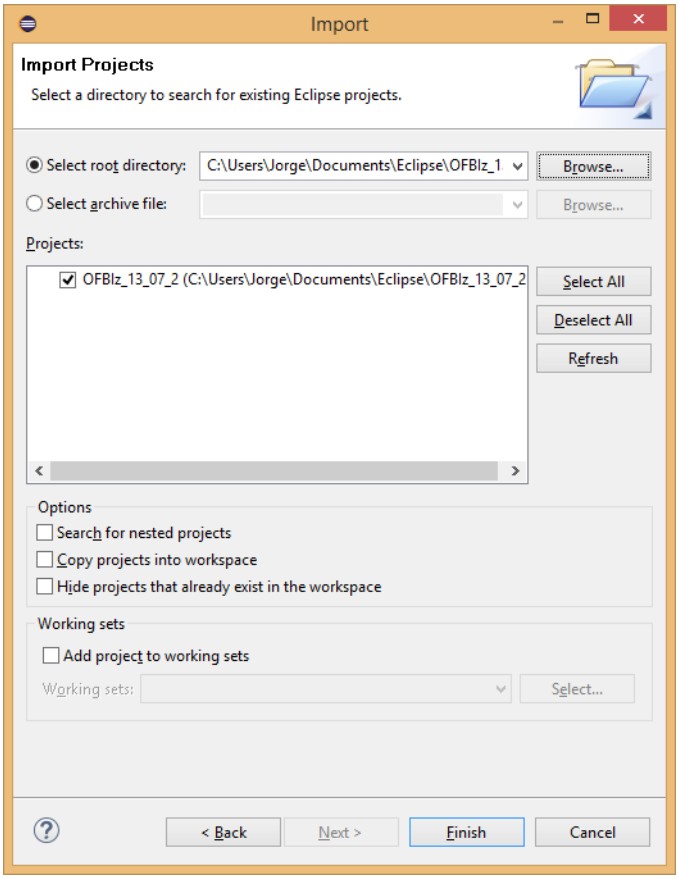


Figure 6 – Selecting the project root directory

Click “Finish” and OFBiz source-code should appear in the Navigatior view.

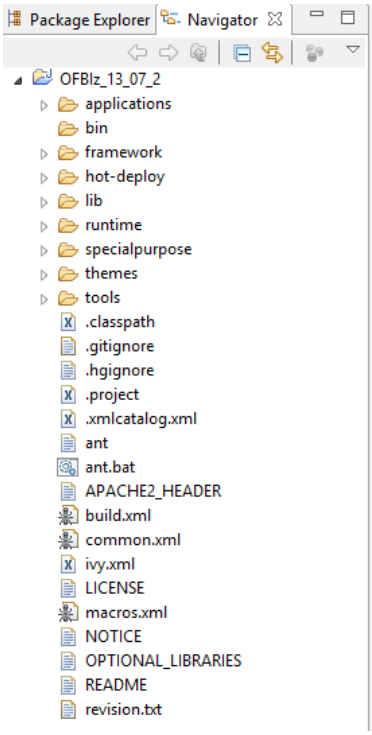


Figure 7 – OFBiz imported project navigator view

Now we can compile the source-code, to do this we need to show the Ant view. To do this go to the "Window" menu > show view and click in the "Ant" menu item.

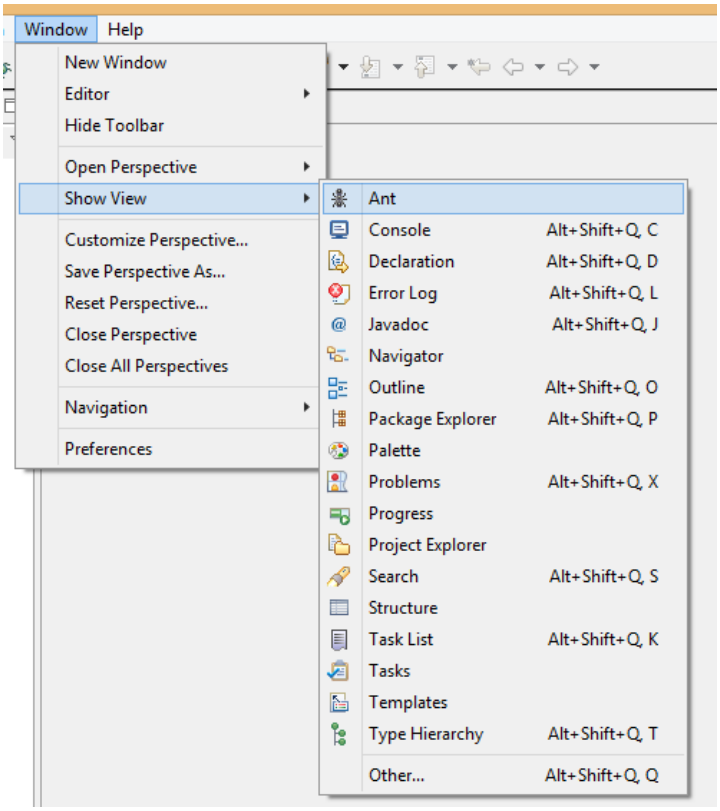


Figure 8 – Showing Eclipse Ant view

With the Ant and Navigator views drag and drop the "build.xml" file from the root folder to the Ant view.

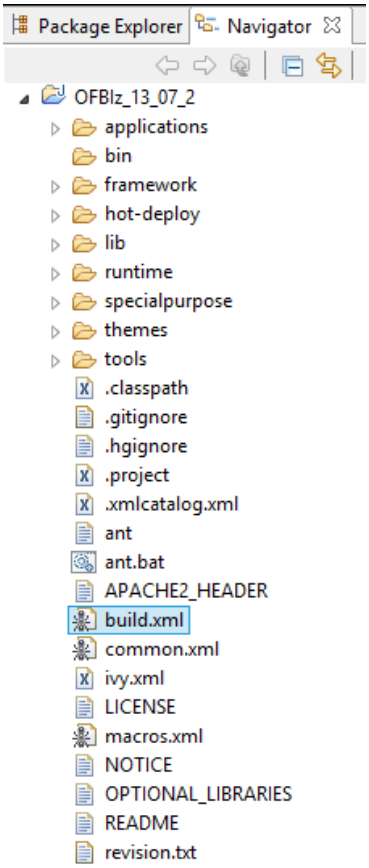


Figure 9 – Highlight of build.xml file in Eclipse

The Ant view should no display all the build options for OFBiz.

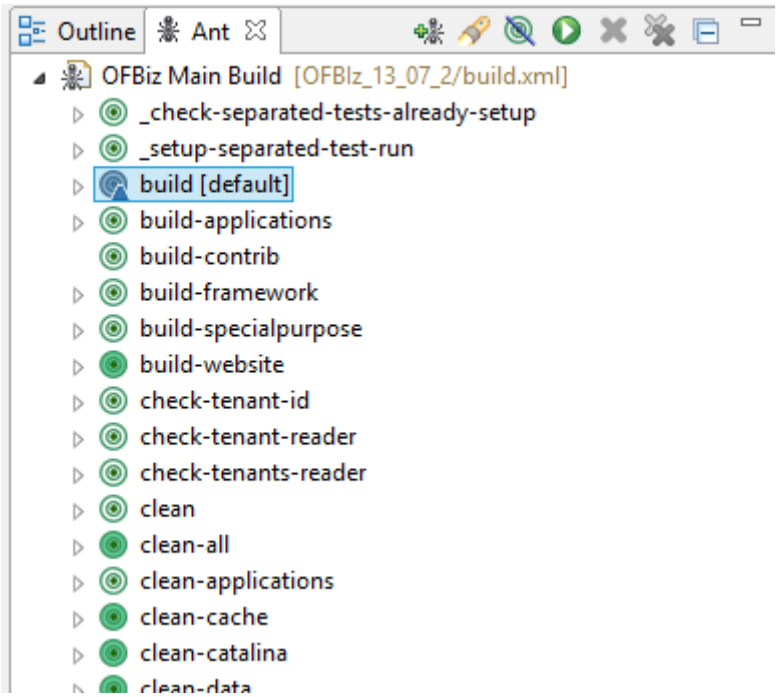


Figure 10 – OFBiz Ant build options

The most important build options are:

- build [default] – Builds OFBiz source code.
- clean – Deletes the result of OFBiz source code compilation
- clean-all – Deletes the result of OFBiz source code compilation and all files created during execution of OFBiz (e.g. logs and databases).
- load-seed – Creates OFBiz database with the essential seed values.
- load-demo – Creates OFBiz database with demo values (this includes, customers, products etc.).

To build OFBiz source code we simply have to double-click on the "build [default]" build option. Note that this may throw a compilation error as follows:

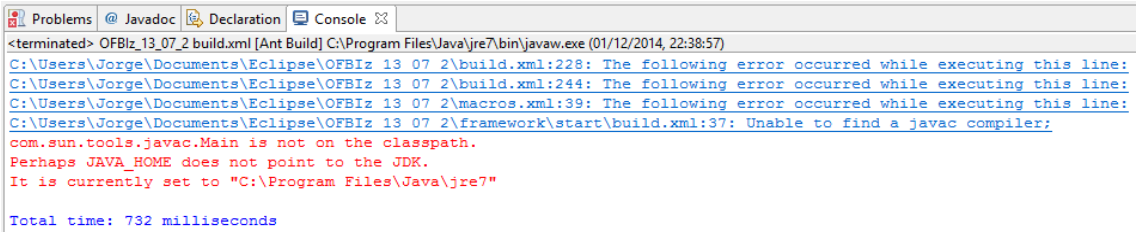


Figure 11 – OFBiz compilation error

This error can be fixed by changing the JRE used to compile, as shown below.

Go to the "Run" menu > External Tools and click the "External Tools Configurations"

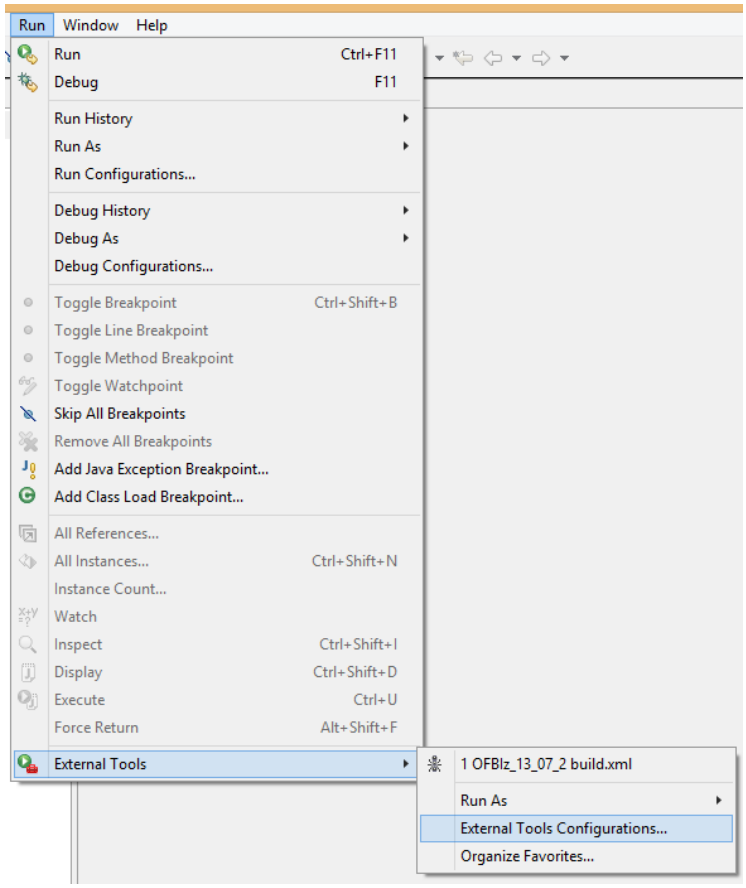


Figure 12 – Eclipse access to External Tools Configurations

Select the Ant build file and go to the JRE tab.

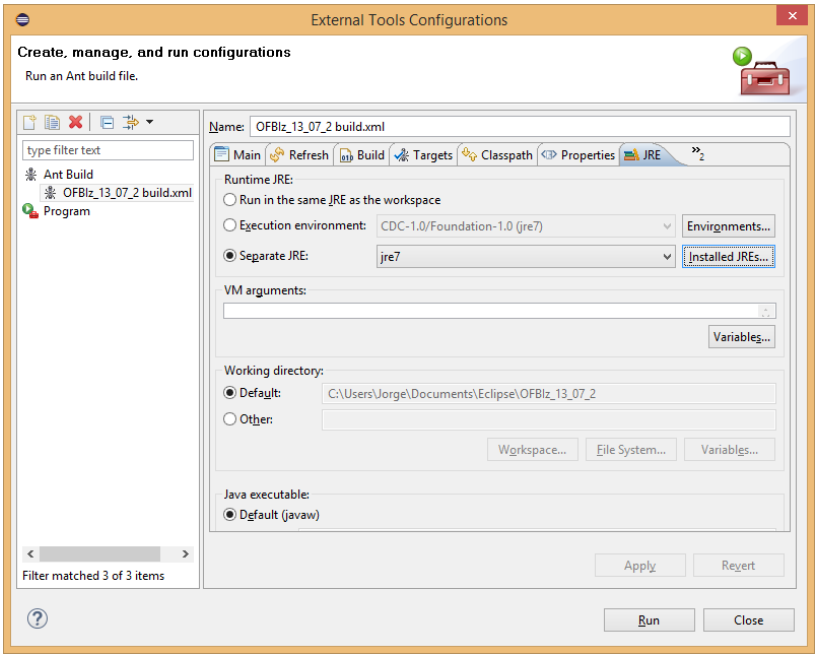


Figure 13 – JRE tab of the external tools configurations

Click in the “Installed JREs” button and then click the Add button.

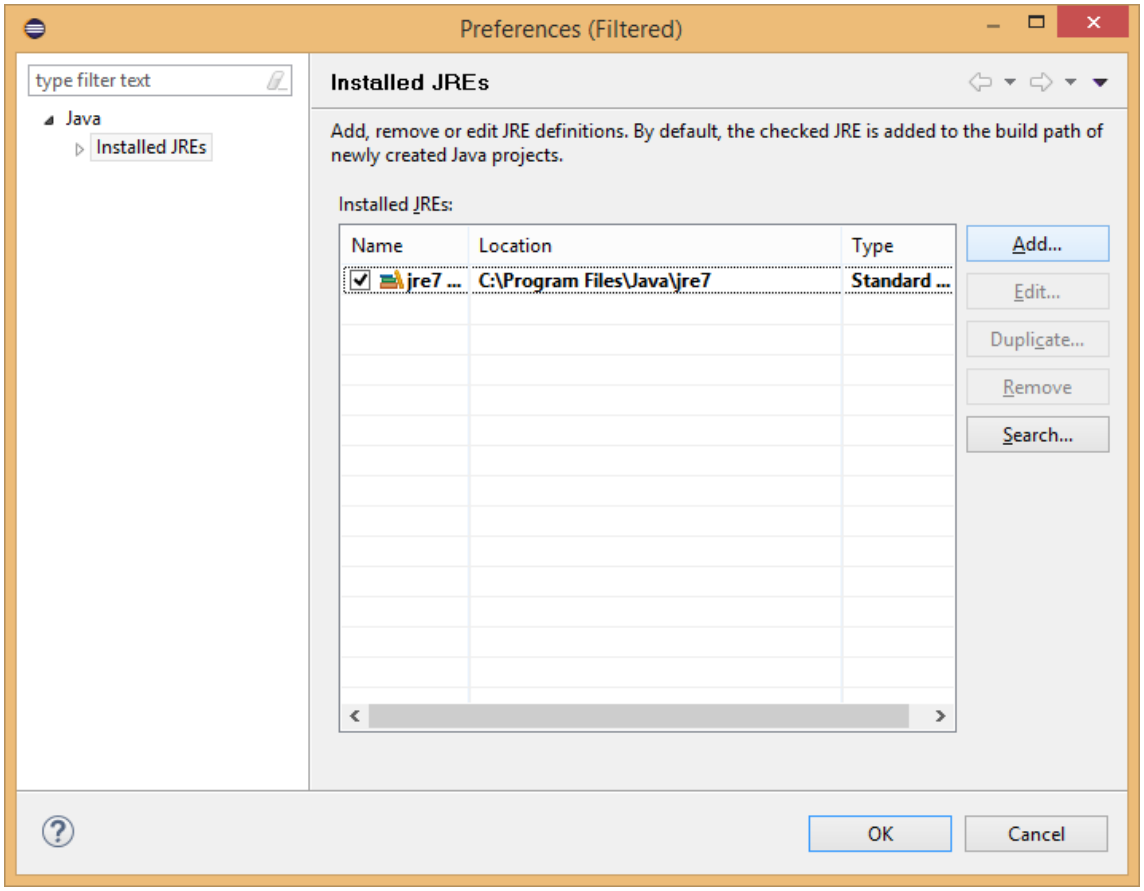


Figure 14 – Installed JREs

Choose the “Standard VM” option.

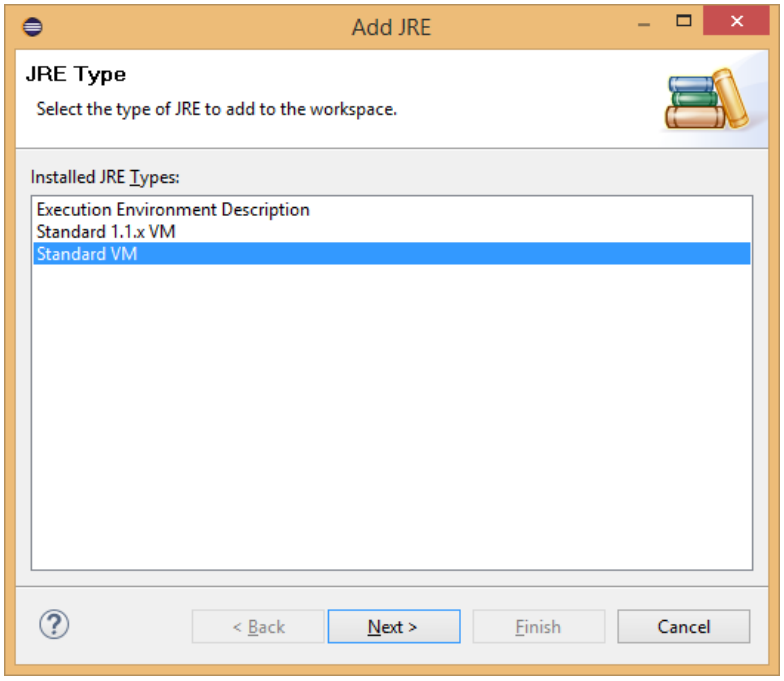


Figure 15 – Selecting a JRE type

Now select the location of the JDK installed on your machine (should be the latest JDK 7).

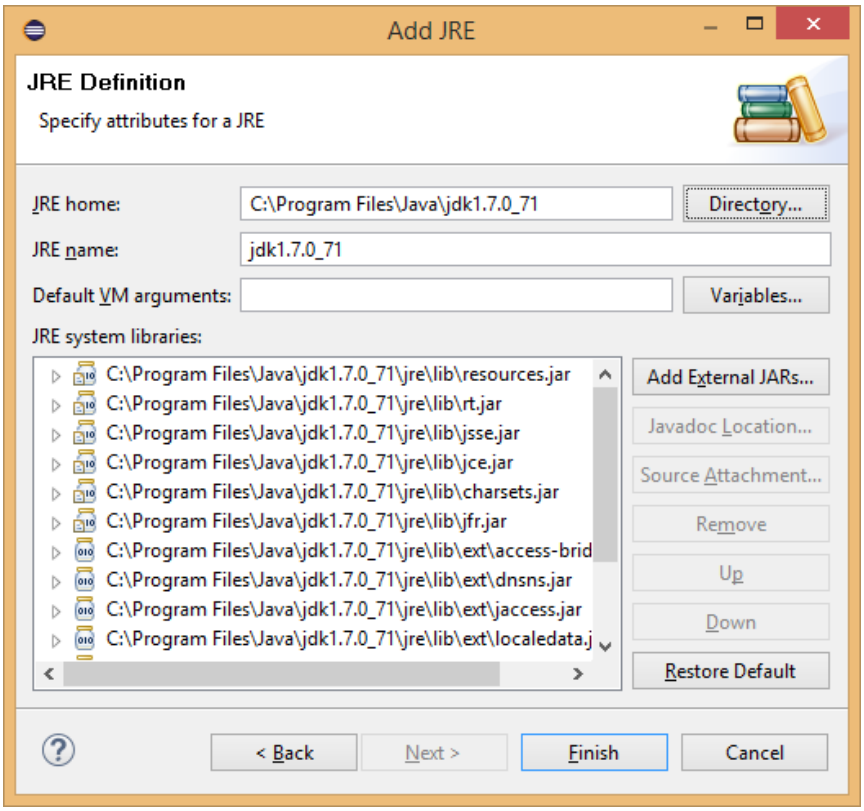


Figure 16 – Choosing the correct JRE

Click Finish and select the newly added JRE defition from the list of Installed JREs.

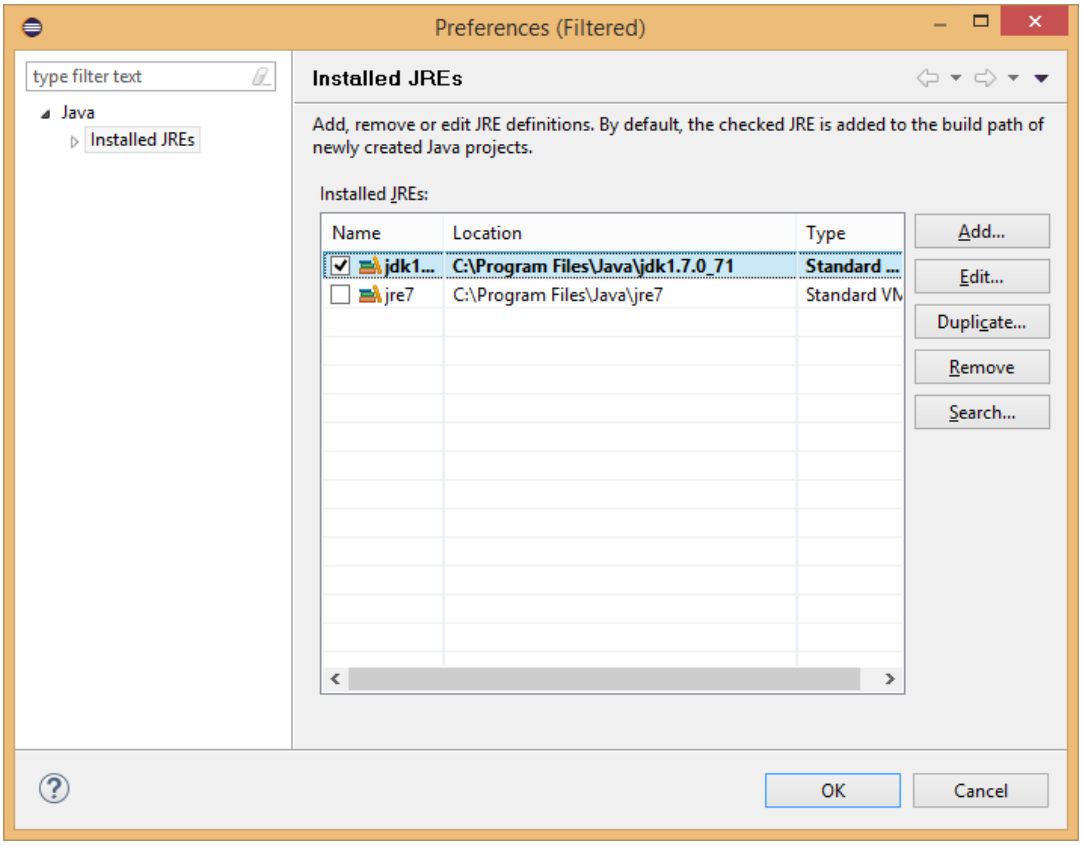


Figure 17 – Changing the JRE selection

Click "Ok". A prompt should appear asking to save the changes made to the Ant build.xml file. Select "Yes"

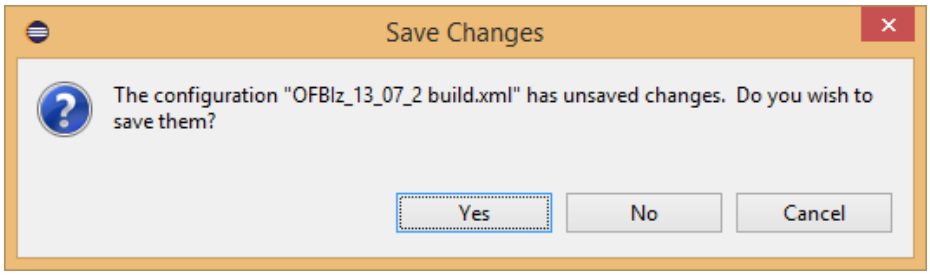


Figure 18 – Saving changes to build.xml file information prompt

Building the solution at this point should present no errors.

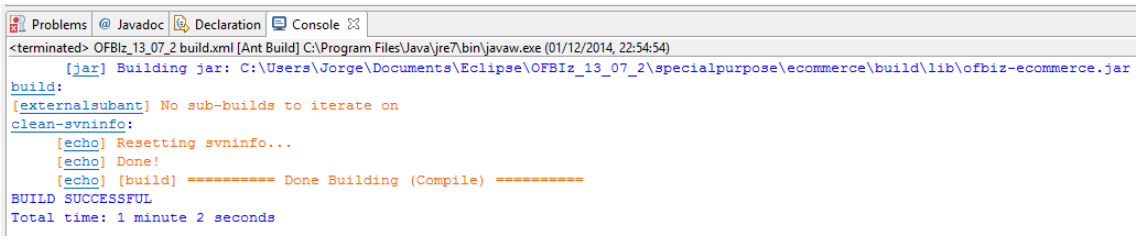


Figure 19 – OFBiz successful compilation message

Click the load-demo Ant definition to create and pre-populate the database with demo values. Note that this step may take a few minutes (5 to 20 minutes depending on the machine where it is running).

2.3 Running OFBiz

OFBiz can be started by executing a simple .bat file. This file is located within the “tools” folder and it is named “startofbiz.bat”. If you execute this bat file you will notice that there is a lot of logging information being written to the console output. For convenience you can change this behavior by writing the outputs that would normally be displayed in the console window to a file. To do this replace the line:

```
"%JAVA_HOME%\bin\java" -Xms128M -Xmx512M -XX:MaxPermSize=512m -jar
ofbiz.jar
```

With:

```
"%JAVA_HOME%\bin\java" -Xms128M -Xmx512M -XX:MaxPermSize=512m -jar
ofbiz.jar > runtime\logs\console.log
```

All the logging contents will be written to the file located at “runtime\logs\console.log”.

You can also declare your Java home location at the beginning of the file, replacing the following line:

```
set OFBIZ_HOME=%~p0..\
```

With:

```
set OFBIZ_HOME=%~p0..\
set JAVA_HOME=C:\Program Files\Java\jdk1.7.0_79\jre\
```

After executing the .bat file OFBiz will be started when the following line is visible in the log file:

```
Connector AJP/1.3 @ 8009 - not-secure
[org.apache.coyote.ajp.AjpProtocol] started.

Connector "http-bio-0.0.0.0-8080" @ 8080 - not-secure
[org.apache.coyote.http11.Http11Protocol] started.

Connector "http-bio-0.0.0.0-8443" @ 8443 - secure
[org.apache.coyote.http11.Http11Protocol] started.

Started Apache Tomcat/7.0.64
```

OFBiz can be accessed using a web browser of your choice and accessing the following address:

```
https://127.0.0.1:8443/catalog/control/main
```

This is the default HTTP endpoint that should be used but there is also a non-secure HTTP endpoint listening at the 8080 port. If you want to use HTTP instead you can access using the following address:

```
https://127.0.0.1:8080/catalog/control/main
```

If you loaded the demo data, the default username is “admin” and the default password is “ofbiz”. This will log you in as the Administrator.

To stop OFBiz simply press CTRL+C buttons.

2.4 Debugging OFBiz

To be able to debug OFBiz we need to change the “startofbiz.bat” to start OFBiz with remote debug capabilities. To enable debugging capabilities replace the following line:

```
"%JAVA_HOME%\bin\java" -Xms128M -Xmx512M -XX:MaxPermSize=512m -jar
ofbiz.jar > runtime\logs\console.log
```

With

```
"%JAVA_HOME%\bin\java" -Xms128M -Xmx512M -XX:MaxPermSize=512m -
Xdebug -Xnoagent -Djava.compiler=NONE -
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005 -jar
ofbiz.jar > runtime\logs\console.log
```

The additions that enable the debugging capabilities are highlighted above. Note that we are creating a remove debug listener on the 5005 port. This will be needed to connect and debug OFBiz.

With OFBiz running, open Eclipse IDE and click the follow button:

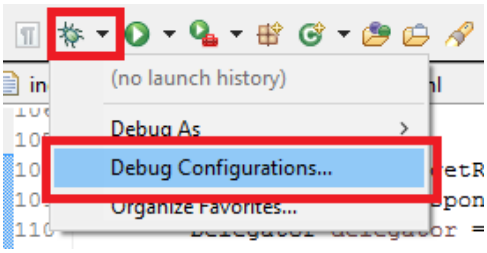


Figure 20 – OFBiz debug button

Click the “Debug Configurations” menu entry and select “Remote Java Application”

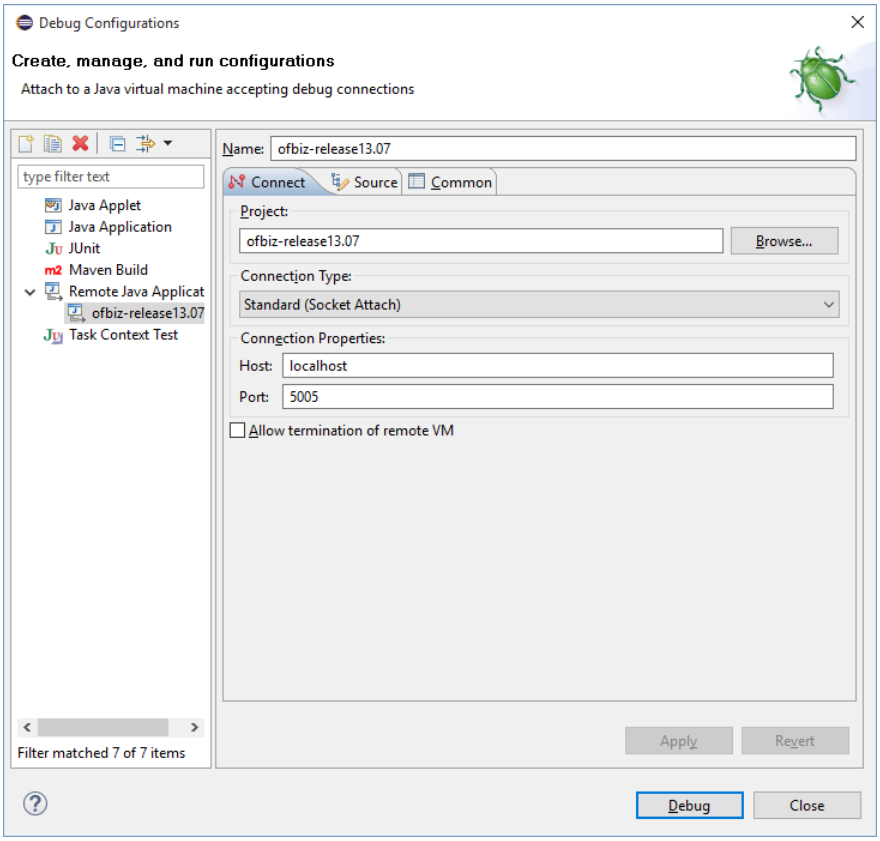


Figure 21 – Remote debug configuration

Change the "Port" field to match the selected debugger port (the one highlighted in the startofbiz.bat file). By default OFBiz uses port 5005.

Change the name field to a friendly name. An entry with this name will appear in the Debug menu allowing you to directly debug OFBiz without having to configure the Port and address again.

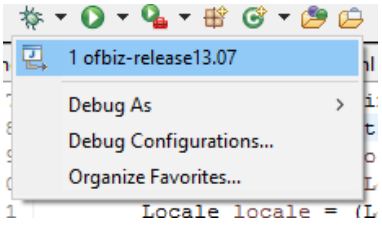


Figure 22 – Remote debug configuration menu entry

Click "Debug" and you will be able to place breakpoints and Debug OFBiz code.

3 MVC Implementation

OFBiz makes use of the MVC (Model-View-Controller) architectural pattern. The model is composed by a set of entities managed by an entity engine. The controller is implemented by a control servlet that will be explained later in detail. At last the views, are implemented as widgets. The following chapters will look in detail into the how the MVC is implemented in OFBiz.

3.1 Entity Engine

The OFBiz entities are the base of the model in the MVC. Each entity is related to one or more database tables, e.g. the person entity defines the fields from the Person table where the information used to describe a person is contained.

"*The Open For Business Entity Engine is a set of tools and patterns used to model and manage entity specific data*" (The Apache Software Foundation, 2015). The primary objective of the entity engine is to eliminate the need to develop entity/service specific persistence code. In order to achieve this level of abstraction, each entity must be configured in the entity engine, specifying the name of the entity, its fields, the type of each field and the relations with other entities. This data access layer abstraction prevents errors on persistence code by enforcing database rules specified in the configuration files and having a generic core capable of dealing with a wide variety of entities and database systems.

3.1.1 Entity Datasources

The entity datasources define, as the name says, the source of the data. By default, OFBiz will use a local Apache Derby[1] relational database management system. These configurations can be found in the following file: "/framework/entity/config/entityengine.xml".

The following xml excerpt contains the default datasource definition for OFBiz 13.07:

[1] https://db.apache.org/derby/

```
<datasource name="localderby"
        helper-class="org.ofbiz.entity.datasource.GenericHelperDAO"
        schema-name="OFBIZ"
        field-type-name="derby"
        check-on-start="true"
        add-missing-on-start="true"
        use-pk-constraint-names="false"
        use-indices-unique="false"
        alias-view-columns="false"
        use-order-by-nulls="true"
        offset-style="fetch">
    <read-data reader-name="tenant"/>
    <read-data reader-name="seed"/>
    <read-data reader-name="seed-initial"/>
    <read-data reader-name="demo"/>
    <read-data reader-name="ext"/>
    <read-data reader-name="ext-test"/>
    <read-data reader-name="ext-demo"/>
    <inline-jdbc
            jdbc-driver="org.apache.derby.jdbc.EmbeddedDriver"
            jdbc-uri="jdbc:derby:ofbiz;create=true"
            jdbc-username="ofbiz"
            jdbc-password-lookup="derby-ofbiz"
            isolation-level="ReadCommitted"
            pool-minsize="2"
            pool-maxsize="250"
            time-between-eviction-runs-millis="600000"/>
</datasource>
```

The datasource element specifies, among other things, the JDBC driver, location, username and password as well as the schema name and isolation levels.

3.1.1.1 Datasource element

The following table describes all the attributes of the datasource element:

Attribute Name	Required?	Description
name	Yes	That name of the datasource.
helper-class	Yes	There can be many types of datasource helpers; the main one used is the JDBC/DAO helper. You can code you own helpers and use them by implementing the **org.ofbiz.entity.GenericHelper** interface. For JDBC/DAO helpers the class will be **org.ofbiz.entity.GenericHelperDAO**.
field-type-name	Yes	The name of the field-type to use. Must match the name of an existing field-type tag as defined above.
schema-name	No	The name of the schema to use in the database. If not specified no schema name will be used. This is a case sensitive value and a different database may expect a different casing.
check-on-start	No	Check the datasource on startup? Must be true or false, defaults to true.
add-missing-on-start	No	Add missing entities and fields to the datasource on startup when checking is done? Must be true or false, defaults to false.
use-foreign-keys	No	Use/Create foreign keys for "one" relationships? Must be true or false, defaults to true.

use-foreign-key-indices	No	Use/Create indices for foreign keys (i.e. an index on the foreign key columns)? Note that creating foreign keys is not required for this to work and that indices are created for type "one" relationship definitions. Must be true or false, defaults to true.
check-fks-on-start	No	Check foreign keys at startup and add missing as needed? Must be true or false, defaults to false. Some databases have a hard time with this and do not return a full list of foreign keys resulting in duplicate foreign keys being added to the database.
check-fk-indices-on-start	No	Check foreign key indices at startup and add missing as needed? Must be true or false, defaults to false.
use-pk-constraint-names	No	Use constraint names for Primary Keys? Some databases have a problem with this, but work fine if they assign their own names. Must be true or false, defaults to true.
constraint-name-clip-length	No	Used to specify max length of a constraint name. Constraint names are clipped to this length. When playing with this watch for duplicate constraint names. Must be an integer, defaults to 30.
fk-style	No	Used to specify the foreign key syntax style, either naming the foreign key constraint, or naming the foreign key itself. Most databases use the name_constraint syntax, but SAP DB is an exception to that and there may be others. Must be either "name_constraint" or "name_fk". Defaults to name_constraint.
use-fk-initially-deferred	No	Used to specify whether or not to use the INITIALLY DEFERRED option available in many databases when creating foreign keys. Not all databases support this option. When enabled and supported the foreign keys will not be checked until a transaction is committed, as opposed to checking foreign keys as operations are done inside a transaction. Must be set to "true" or "false". Defaults to true.
join-style	No	Used to specify the syntax to use when doing table joins in view-entity operations. Many databases are adopting the ANSI JOIN standard, but before that was introduced theta joins were much more common. Two theta join styles are supported: Oracle and MS SQL. Must be "ansi", "ansi-no-parenthesis", "theta-oracle" or "theta-mssql". Defaults to "ansi".
use-indices	No	Used to specify whether or not declared indices/indexes should be created in the database. Must be "true" or "false", defaults to "true".
check-indices-on-start	No	Used to specify whether or not indices/indexes should be checked when the server starts. Will add missing declared indexes. Note that many JDBC drivers do not support this sort of meta-data check and this may cause problems. Must be "true" or "false", defaults to "false".
alias-view-columns	No	This is used to compensate for a variation seen in some JDBC drivers where column names returned for aliased fields (especially in view entities) be either be the alias name, or the full text of what makes up that alias name. Most

		databases return the alias name, so this will generally be set to true. Must be "true" or "false", defaults to "true".

Table 1 – Datasource element attributes

The following table provides a description of each of the supported elements within the Datasource element:

Sub-Element Name	How Many	Description
ANY	0 or 1	Any tag may go inside the datasource tag to specify parameters for other GenericHelper implementations. These will not be checked at load time unless the DTD is modified to describe them.
inline-jdbc	0 or 1	Used to specify the JDBC parameters to be used either by the connection pool or if no connection pool is available then by directly loading the driver (very slow). You must specify either inline-jdbc or jndi-jdbc, but not both, for the datasource.
jndi-jdbc	0 or 1	Used to specify the jndi-server and jndi-name to get a Connection or XAConnection from JNDI. You must specify either inline-jdbc or jndi-jdbc, but not both, for the datasource.
sql-load-path	0 to many	Used to specify a list of full paths to directories that will be searched for XML and SQL files to import into the data source by the **install** page in the WebTools webapp. Each tag has two attributes: path for the path location, and prepend-env to optionally specify a Java environment property to prepend to the specified path.

Table 2 – Sub-elements of the Datasource element

The following table describes the attributes of the inline-jdbc element:

Attribute Name	Required?	Description
jdbc-driver	Yes	The JDBC driver class for the database.
jdbc-uri	Yes	The URI used to specify the type and location of the database.
jdbc-username	Yes	The username to connect to the database as.
jdbc-password	Yes	The username's password.
isolation-level	No	This is used by Tyrex to specify the transaction isolation level. The standard JDBC transaction isolation levels are available: • "None" • "ReadCommitted" • "ReadUncommitted" • "RepeatableRead" • "Serializable"

Table 3 – Inline-jdbc element attributes

The following table describes the attributes of the jndi-jdbc element:

Attribute Name	Required?	Description
jndi-server-name	Yes	The name of the JNDI Server to use as configured in this file with the jndi-server tag, described above.
jndi-name	Yes	The name of the Connection or XAConnection object in JNDI.

Table 4 – Jndi-jdbc element attributes

3.1.2 Entity Delegators

OFBiz uses entity delegators to access the database. The delegators provide means to execute CRUD operations to the database but do not execute these operations themselves, instead the delegator will look for the datasource of the entity group in which the entity being accessed is contained and delegate the operation to the specific datasource. The delegators are defined in the same file as the datasources. The following xml excerpt shows the default OFBiz delegator:

```
<delegator name="default" entity-model-reader="main"
      entity-group-reader="main" entity-eca-reader="main"
      distributed-cache-clear-enabled="false">
        <group-map group-name="org.ofbiz"
            datasource-name="localderby"/>
        <group-map group-name="org.ofbiz.olap"
            datasource-name="localderbyolap"/>
        <group-map group-name="org.ofbiz.tenant"
            datasource-name="localderbytenant"/>
</delegator>
```

3.1.2.1 Delegator element

The following table provides documentation for all the supported attributes of the delegator element:

Attribute Name	Required?	Description
distributed-cache-clear-class-name	No	Used to specify if the name of the class to use for DCC and that implements the distributed cache clear interface. If not specified defaults to "org.ofbiz.entityext.cache.EntityCacheServices", which is a good default for most cases that uses the Service Engine for configuration and remote calls.
distributed-cache-clear-enabled	No	Used to specify if the distributed cache clear should be enabled. If not specified defaults to "false". If set to true the other DCC attributes will be used.
distributed-cache-clear-user-login-id	No	Used to specify if the userLoginId (to use for any security checks) is needed related to distributed cache clear operation. If not specified defaults to "admin".
entity-eca-reader	No	The name of the entity-eca-reader to use for this delegator. If not specified no Entity ECAs will be used.
entity-group-reader	Yes	The name of the entity-group-reader to use for this delegator.
entity-model-reader	Yes	The name of the entity-model-reader to use for this delegator.
name	Yes	The name of the Delegator. Used to look up this tag by delegator name.

Table 5 – Delegator element attributes

The following table describes all the elements supported by the delegator element:

Sub-element name	How many	Description
Group-map	0 to many	Used to specify the datasource name where the entities associated with this group are stored.

Table 6 – Sub-elements of the Delegator element

Group-map sub-element

The following table provides documentation for all the supported attributes of the group-map element:

Attribute name	Required?	Description
Group-name	Yes	Name of the ofbiz entity group.
Datasource-name	Yes	Name of the datasource.

Table 7 – Group-map element attributes

3.1.3 Entity Groups

An entity group is, as the name indicates, a named group that contains multiple entities that are located on the same datasource. This name is used by the entity engine delegators to decide which datasource contains which entity. The main entity group of OFBiz is called “org.ofbiz”, by default all entities are associated with this group.

The association between entities and entity group are declared in the entitygroup*.xml file inside the entitydef folder under each component.

```
<entity-group group="org.ofbiz.olap"
      entity="SalesInvoiceItemFact"/>
```

3.1.4 Entity definition

In OFBiz there is no need to create new java classes whenever we need to create a new database entity and use it in the application. To accomplish this, each entity is configured in a XML file that contains the fields, field types, primary keys, foreign keys, and so on, for each entity. Because there is no java class associated with an entity, the framework needs to be smart and generic enough to support all possible entities, as such, the manipulation of an entity on the application side is made using generic objects such as generic maps. These generic maps provide access to all database fields by name.

The entities are configured in the /entitydef/entitymodel*.xml file within each application component. The following XML excerpt shows the definition of the Party entity.

```
<entity entity-name="Party" package-name="org.ofbiz.party.party"
  title="Party Entity">
  <field name="partyId" type="id-ne"></field>
  <field name="partyTypeId" type="id-ne"></field>
  <field name="externalId" type="id"></field>
  <field name="preferredCurrencyUomId" type="id-ne"></field>
  <field name="description" type="very-long"></field>
  <field name="statusId" type="id-ne"></field>
  <field name="createdDate" type="date-time"></field>
  <field name="createdByUserLogin" type="id-vlong"></field>
  <field name="lastModifiedDate" type="date-time"></field>
  <field name="lastModifiedByUserLogin" type="id-vlong"></field>
  <field name="dataSourceId" type="id"></field>
  <field name="isUnread" type="indicator"></field>
  <prim-key field="partyId" />
  <relation type="one" fk-name="PARTY_PTY_TYP"
      rel-entity-name="PartyType">
    <key-map field-name="partyTypeId" />
  </relation>
  <relation type="one" fk-name="PARTY_CUL" title="CreatedBy"
    rel-entity-name="UserLogin">
    <key-map field-name="createdByUserLogin"
      rel-field-name="userLoginId" />
  </relation>
  <relation type="one" fk-name="PARTY_LMCUL" title="LastModifiedBy"
    rel-entity-name="UserLogin">
    <key-map field-name="lastModifiedByUserLogin"
      rel-field-name="userLoginId" />
  </relation>
  <relation type="one" fk-name="PARTY_PREF_CRNCY"
    rel-entity-name="Uom">
    <key-map field-name="preferredCurrencyUomId"
      rel-field-name="uomId" />
  </relation>
  <relation type="one" fk-name="PARTY_STATUSITM"
    rel-entity-name="StatusItem">
    <key-map field-name="statusId" />
  </relation>
  <relation type="many" rel-entity-name="PartyTypeAttr">
    <key-map field-name="partyTypeId" />
  </relation>
  <relation type="one" fk-name="PARTY_DATSRC"
    rel-entity-name="DataSource">
    <key-map field-name="dataSourceId" />
  </relation>
  <index name="PARTYEXT_ID_IDX">
    <index-field name="externalId" />
  </index>
</entity>
```

3.1.4.1 Entity element

The following table describes the attributes of the entity element:

Attribute Name	Required?	Description
entity-name	Yes	The name of the entity as it is referred to when using the Entity Engine Java API and various other places in the Entity Engine.
table-name	No	The name of the database table that corresponds to this entity. This attribute is optional and if not specified the table name will be derived from the entity name.
package-name	Yes	The name of the package that this entity is contained in. With hundreds of entities in a large data model this is used to organize and structure the entities definitions.
dependent-on	No	This can be used to specify a parent entity or an entity that this entity is dependent on. This is currently not used for anything automated in the Entity Engine, but can be used to specify an heirarchical entity structure.
enable-lock	No	Specifies whether or not optimistic locking should be used for this entity. The **lastUpdatedStamp** field must exist on the entity and will be used to keep track of the last time the entity instance was updated. If the current instance to be updated does not have a matching lastUpdatedStamp an EntityLockedException will be thrown. Must be true or false. Defaults to false.
never-cache	No	If this is set to true caching of this entity will not be allowed. Automatic cache clearing will not be done to improve efficiency and any attempt to use the cache methods on the entity will result in an exception so that it is easier to find and eliminate where this is being done. Must be true or false. Defaults to false.
title	No	A title for the entity. If not specified defaults to the global setting for the file the entity is in.
copyright	No	The copyright of the entity. If not specified defaults to the global setting for the file the entity is in.
author	No	The author of the entity. If not specified defaults to the global setting for the file the entity is in.
version	No	The version of the entity. If not specified defaults to the global setting for the file the entity is in.

Table 8 – Entity element attributes

The following table describes the sub-elements of the entity element.

Sub-Element Name	How Many	Description
description	0 or 1	A description of the entity. If not specified defaults to the global setting for the file the entity is in. This element has no attributes and should contain only a simple string of characters.
field	1 to many	Used to declare fields that are part of the entity.
prim-key	0 to many	Used to declare which fields are primary keys
relation	0 to many	Used to declare relationships between entities.

Table 9 – Sub-elements of the Entity element

Field sub-element

The following table describes all attributes of the field element:

Attribute Name	Required?	Description
name	Yes	The name of the field that is used to refer to it in Java code and other places.
col-name	No	The name of the corresponding database column. This is not required and if not specified this will be derived from the field name.
type	Yes	The type of the field. This is looked up in the field types file for the current datasource at run-time to determine the Java and SQL types for the field and database column.

Table 10 – Field element attributes

Sub-Element Name	How Many	Description
validate	0 to many	Each validate element has a single attribute called name which specifies the name of the validation method to call. These methods are not called in all Entity Engine operations and are only used for generic user interfaces like the Entity Data Maintenance pages in WebTools.

Table 11 – Sub-elements of the Field element

Prim-key sub-element

Attribute Name	Required?	Description
field	Yes	The name of the field that will be part of the primary key.

Table 12 – Prim-key element attributes

Relation sub-element

Attribute Name	Required?	Description
type	Yes	Specifies the type of the relationship including the cardinality of the relationship (in one direction) and if a foreign key should be created for cardinality one relationships. Must be "one", "one-nofk", or "many".
title	No	Because you may want to have more than one relationship to a single entity this attribute allows you to specify a title that will be prepended to the rel-entity-name to make up the name of the relationship. If not specified the rel-entity-name alone will be used as the relationship name.
rel-entity-name	Yes	The name of the related entity. The relationship goes from this entity to the related entity.
fk-name	No	The foreign key name can be created automatically from the relationship name, but this is not recommended for two reasons: many databases have a very small maximum size (like 18 characters) for foreign key and index names, and many databases require that the FK name be unique for the entire database and not just for the table the FK is coming from.

Table 13 – Relation element attributes

Sub-Element Name	How Many	Description
key-map	1 to many	The key-map is used to specify a field in this entity that corresponds to a field in the related entity. This element has two attributes: **field-name** and **rel-field-name**. These are used to specify the name of the field on this entity and the corresponding name of the field on the related entity.

Table 14 – Sub-elements of the Relation element

3.1.5 View Entities

In addition to the OFBiz entities that map the fields of a single database table, OFBiz also supports View Entities. View Entities allow the creation of a "virtual" entity, composed by fields of one or more database tables, similar to what can be achieved by using a "View" in Oracle Database or Microsoft SQL Server. The fields of a View entity will be alias of the original fields and can be either directly fetched from one of the view entities or calculated. View entities are defined in the same location as the entities and need to be associated with an entity group. The XML excerpt below shows the definition of the PartyAndGroup view entity.

```
<view-entity entity-name="PartyAndGroup"
      package-name="org.ofbiz.party.party"
      title="Party and Party Group View Entity">
      <member-entity entity-alias="PTY" entity-name="Party" />
      <member-entity entity-alias="PGRP"
            entity-name="PartyGroup" />
      <alias-all entity-alias="PTY" />
      <alias-all entity-alias="PGRP" />
      <view-link entity-alias="PTY" rel-entity-alias="PGRP">
            <key-map field-name="partyId" />
      </view-link>
</view-entity>
```

This view would generate an SQL as the one presented below:

```
SELECT
    PTY.PREFERRED_CURRENCY_UOM_ID,
    PTY.PARTY_TYPE_ID,
    PTY.EXTERNAL_ID,
    PTY.DATA_SOURCE_ID,
    PTY.IS_UNREAD,
    PTY.STATUS_ID,
    PTY.LAST_MODIFIED_BY_USER_LOGIN,
    PTY.PARTY_ID,
    PTY.DESCRIPTION,
    PTY.LAST_MODIFIED_DATE,
    PTY.CREATED_DATE,
    PTY.CREATED_BY_USER_LOGIN,
    PGRP.GROUP_NAME,
    PGRP.LOGO_IMAGE_URL,
    PGRP.GROUP_NAME_LOCAL,
    PGRP.TICKER_SYMBOL,
    PGRP.OFFICE_SITE_NAME,
    PGRP.ANNUAL_REVENUE,
    PGRP.NUM_EMPLOYEES,
    PGRP.COMMENTS
FROM OFBIZ.PARTY PTY
INNER JOIN OFBIZ.PARTY_GROUP PGRP ON PTY.PARTY_ID = PGRP.PARTY_ID
ORDER BY PTY.PARTY_ID ASC
```

The view definition presented above defines an entity that joins the Party entity with the PartyGroup entity and selects all the resulting fields. In this case, as both can entities probably have fields with the same name we define the alias of “PTY” for the Party entity and “PGRP” for the PartyGroup entity. In short, the member-entity element defines what entities the view will use and what aliases they will have, the alias-all indicates that all fields from the entity with the provided alias will be selected and the view-link defines how the entities are combined, in this case they will be combined by the values of the field named “partyId”.

3.1.5.1 Grouping or Summary data

In SQL it is possible to group data by a given criteria or have fields that are the result of a given function or the combination of both. In OFBiz this can be done by using View Entities with grouping criteria and/or functions. Grouping a set of results by a given criteria is configured using the group-by sub-element of the view-entity element.

```
<view-entity entity-name="CommunicationEventSum"
  package-name="org.ofbiz.party.communication"
  title="Sum of communication events over status">
  <member-entity entity-alias="CE"
      entity-name="CommunicationEvent" />
  <alias entity-alias="CE" name="communicationEventId"
      function="count" />
  <alias entity-alias="CE" name="statusId" group-by="false" />
  <alias entity-alias="CE" name="partyIdTo" group-by="true" />
</view-entity>
```

The entity described above presents a sum of communication events by Party Id. The function attribute specifies what value will be presented for the grouped values.

Regarding the use of functions as the value of a table field, OFBiz currently supports the following functions:

- min
- max
- sum
- avg
- count
- count-distinct
- upper
- lower

Not all of the above functions require a group by to be used, they can be used as the output of the fields on view entities. These functions can be used as follows:

```
<view-entity entity-name="ProductVirtualAndAssocPrices"
  package-name="org.ofbiz.product.product" title="Virtual and
Associated Product Prices View Entity">
  <member-entity entity-alias="PVIRT" entity-name="Product" />
  <member-entity entity-alias="PA" entity-name="ProductAssoc" />
  <member-entity entity-alias="PASC" entity-name="Product" />
  <member-entity entity-alias="PASCPRC"
      entity-name="ProductPrice" />
  <alias entity-alias="PVIRT" name="productId" />
  <alias entity-alias="PVIRT" name="internalName" />
  <alias entity-alias="PVIRT" name="productName" />
  <alias entity-alias="PA" name="productAssocTypeId" />
  <alias entity-alias="PA" name="fromDate" />
  <alias entity-alias="PA" name="thruDate" />
  <alias entity-alias="PASC" name="assocProductId"
      field="productId" />
  <alias entity-alias="PASC" name="assocProductCount"
      field="productId"
    function="count-distinct" />
  <alias entity-alias="PASCPRC" name="assocPriceTypeId"
      field="productPriceTypeId" />
  <alias entity-alias="PASCPRC" name="assocCurrencyUomId"
      field="currencyUomId" />
  <alias entity-alias="PASCPRC" name="assocProductStoreGroupId"
    field="productStoreGroupId" />
  <alias entity-alias="PASCPRC" name="assocPriceFromDate"
      field="fromDate" />
  <alias entity-alias="PASCPRC" name="assocPriceThruDate"
      field="thruDate" />
  <alias entity-alias="PASCPRC" name="assocMinPrice" field="price"
    function="min" />
  <alias entity-alias="PASCPRC" name="assocMaxPrice" field="price"
    function="max" />
  <view-link entity-alias="PVIRT" rel-entity-alias="PA">
    <key-map field-name="productId" />
  </view-link>
  <view-link entity-alias="PA" rel-entity-alias="PASC">
    <key-map field-name="productIdTo" rel-field-name="productId" />
  </view-link>
  <view-link entity-alias="PASC" rel-entity-alias="PASCPRC">
    <key-map field-name="productId" />
  </view-link>
</view-entity>
```

3.1.6 Complex Aliases

The functions presented above do not address all of the common cases, for example we have not yet seen how it is possible to sum two fields. Complex aliases add support for operations that involve two or more entity fields, e.g. fieldA + fieldB.

Complex aliases are defined by using the complex-alias sub-element of the alias element and it is possible to nest complex aliases. The following examples shows how to subtract two fields:

```
<alias entity-alias="OI" name="quantityOrdered" function="sum">
  <complex-alias operator="-">
    <complex-alias-field entity-alias="OI" field="quantity"
      default-value="0" />
    <complex-alias-field entity-alias="OI" field="cancelQuantity"
      default-value="0" />
  </complex-alias>
</alias>
```

The following example shows how to nest complex aliases in order to retrieve the result of the following equation "(a + b) * ((c - d) / e)":

```
<alias name="complexComputedField">
  <complex-alias operator="*">
    <complex-alias operator="+">
      <complex-alias-field entity-alias="EA" field="a"/>
      <complex-alias-field entity-alias="EA" field="b"/>
    </complex-alias>
    <complex-alias operator="/">
      <complex-alias operator="-">
        <complex-alias-field entity-alias="EA" field="c"/>
        <complex-alias-field entity-alias="EA" field="d"/>
      </complex-alias>
      <complex-alias-field entity-alias="EA" field="e"/>
    </complex-alias>
  </complex-alias>
</alias>
```

The complex alias operator is not database independent, meaning that whatever is placed in the operator will be processed as is and might not work on all databases supported by OFBiz.

3.1.7 Extended Entities

Extending entities can be useful when we need to add a new field onto an existing entity but do not want to change OFBiz original entity definitions (which would cause some problems when upgrading to newer versions). The following example extends the entity “Visitor” with the “partyId” field:

```
<extend-entity entity-name="Visitor">
  <field name="partyId" type="id"></field>
  <relation type="one" fk-name="VISITOR_PARTY"
      rel-entity-name="Party">
    <key-map field-name="partyId" />
  </relation>
</extend-entity>
```

3.1.8 Database Independence

OFBiz is database independent, with a few exceptions. As the data types of table fields vary from database to database, OFBiz has its logic attached to OFBiz data types that can be considered as “virtual” data types that will be converted to the database specific data type at runtime. The configuration of the specificities of each supported database system can be found under the /framework/entity/fieldtype/.

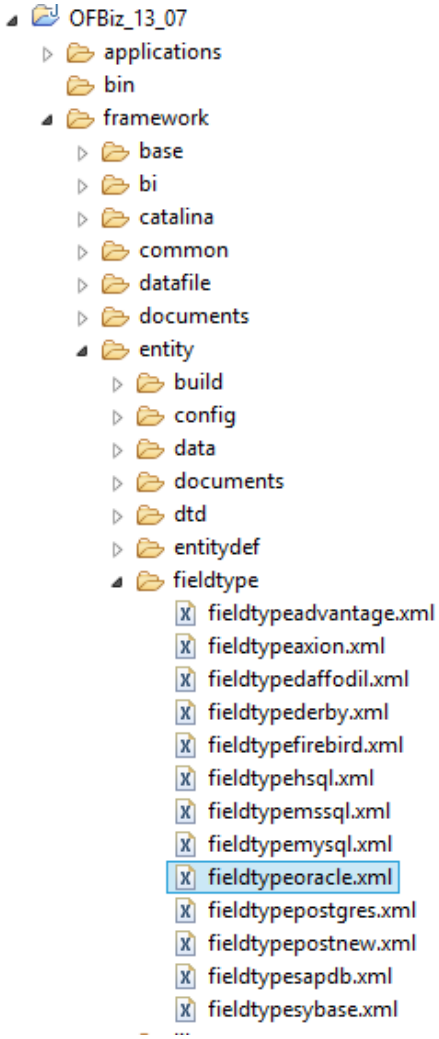


Figure 23 – Database field configuration

Inside the fieldtype folder there is one XML file for each database system supported by OFBiz. Each file contains the specifications of each OFBiz type, database type and java type. The XML displayed below shows the field type configuration for Oracle DBMS.

```
<field-type-def type="blob"
      sql-type="BLOB" java-type="java.sql.Blob"/>
<field-type-def type="byte-array"
      sql-type="BLOB" java-type="byte[]"/>
<field-type-def type="object"
      sql-type="BLOB" java-type="Object"/>
<field-type-def type="date-time"
      sql-type="TIMESTAMP" sql-type-alias="TIMESTAMP(6)"
      java-type="java.sql.Timestamp"/>
<field-type-def type="date"
      sql-type="DATE" java-type="java.sql.Date"/>
<field-type-def type="time"
      sql-type="DATE" java-type="java.sql.Time"/>
<field-type-def type="url"
      sql-type="VARCHAR2(255)" java-type="String"/>
<field-type-def type="id-ne"
      sql-type="VARCHAR2(20)" java-type="String"/>
<field-type-def type="id-long-ne"
      sql-type="VARCHAR2(60)" java-type="String"/>
```

3.2 The Control Servlet

The controller in OFBiz is called the control servlet. The control servlet is responsible for listening and responding to all incoming requests from the end-user. The control servlet acts a connection between the entity engine, service engine and presentation layer. The control servlet implements the front controller pattern.

The following list enumerates a set of common tasks that are executed by the control servlet:

- Perform security checks;
- Log the request;
- Perform pre-processing;
- Lookup the defined processing for the request;
- Process events, if any;
- Determine the response and render it.

Each module of the OFBiz has its own control servlet defined in the web.xml file. The Web.xml can be broken into the following configurations:

- A name and description:

```
<display-name>Open For Business - Party Manager</display-name>
<description>Party Manager Module of the Open For Business
Project</description>
```

- What parameters will be available for the controllers, views and services to use:

```
<context-param>
  <param-name>entityDelegatorName</param-name>
  <param-value>default</param-value>
  <description>The Name of the Entity Delegator to use, defined in
entityengine.xml</description>
</context-param>
<context-param>
  <param-name>localDispatcherName</param-name>
  <param-value>partymgr</param-value>
  <description>A unique name used to identify/recognize the local
dispatcher for the Service Engine</description>
</context-param>
<context-param>
  <param-name>mainDecoratorLocation</param-name>
  <param-value>
      component://party/widget/partymgr/CommonScreens.xml
  </param-value>
  <description>The location of the main-decorator screen to use for
      this webapp; referred to as a context variable in screen def
      XML files.
  </description>
</context-param>
<context-param>
  <param-name>scriptLocationPath</param-name>
  <param-value>/WEB-INF/bsh</param-value>
  <description>BeanShell Script Location</description>
</context-param>
```

These parameters are called context parameters. Most of the application modules will configure at least the following parameters:

- Entity delegator – used for database access;
- Local service dispatcher – used to invoke services;
- Location of the main screen decorator – used to decorate all screens of the module;
- Location of BeanShell scripts. BeanShell scripts are currently deprecated in favor of Groovy scripts so this parameter is rarely used;

- What files can be served to users:

```
<filter>
  <filter-name>ContextFilter</filter-name>
  <display-name>ContextFilter</display-name>
  <filter-class>org.ofbiz.webapp.control.ContextFilter</filter-
class>
  <init-param>
    <param-name>disableContextSecurity</param-name>
    <param-value>N</param-value>
  </init-param>
  <init-param>
    <param-name>allowedPaths</param-name>
    <param-value>
      /error:/control:/select:/index.html:/index.jsp:/default.html:
      /default.jsp:/images:/static:/js
    </param-value>
  </init-param>
  <init-param>
    <param-name>errorCode</param-name>
    <param-value>403</param-value>
  </init-param>
  <init-param>
    <param-name>redirectPath</param-name>
    <param-value>/control/main</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>ContextFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

This is called context security filter and it's used to restrict access to certain application files. By default, the access to all files is rejected until it is specifically defined that a file or folder can be requested. The list of allowed files and folders is defined inside an init-param element whose param-name inner element value is "allowedPaths".

In the example above the following is permitted:

- Access to all files inside the following folders: /error, /control, /select, /images, /static, /js. Note that the "control" folder does not exist physically, but it is part of an expected URL pattern.
- Access to the /index.html, /index.jsp, /default.html, /default.jsp.

If a file that is not permitted is requested the server can either return an HTTP error with the error code specified in the filter parameter named "errorCode", or it can redirect to the page specified in the filter parameter named "redirectPath".

- Implementation of event listeners and control servlets:

```
<listener>
  <listener-
class>org.ofbiz.webapp.control.ControlEventListener</listener-
class>
</listener>
<listener>
  <listener-
class>org.ofbiz.webapp.control.LoginEventListener</listener-class>
</listener>
<servlet>
  <servlet-name>ControlServlet</servlet-name>
  <display-name>ControlServlet</display-name>
  <description>Main Control Servlet</description>
  <servlet-class>org.ofbiz.webapp.control.ControlServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

- What URLs should be intercepted by the control servlet:

```
<servlet-mapping>
  <servlet-name>ControlServlet</servlet-name>
  <url-pattern>/control/*</url-pattern>
</servlet-mapping>
```

In this case only requests that start with /control/ will be intercepted by OFBiz framework, this means that requests that do not match this pattern will be handled by the default server. OFBiz uses an embedded Apache Tomcat server by default. This servlet mapping is the requirement that led to the definition of the /control in the allowed paths of the context security filter.

The web.xml file explained above is only the entry point of the control servlet and does not contain the actual URLs that will be served and what will actually be done. This is specified in the controller.xml file. This file contains the requests that the module is capable of responding to, what security constraints must be met in order to execute the request logic, what events are fired, what "commands" need to be executed and what View should be written as output.

The possible elements within a controller element must appear in the following order:

- Include;
- Description;
- Handler;
- Request-map;
- View-map.

3.2.1 Controller configurations

OFBiz controllers are responsible for handling all OFBIz requests, it's what contains the logic of the web application. A controller defines what requests can be made, what responses can be sent to the client and who is responsible for building the response.

3.2.1.1 Handlers

Handlers are the entities responsible for building a certain type of response, e.g. a view (which will generate HTML). There are two types of handlers in OFBiz:

- View Handlers – Responsible for rendering a specific view to html;
- Event handlers – Responsible for triggering events.

The most commonly used view handlers are:

Handler name (s)	Handler class	Description
screen **screenxml** **screentext** **screencsv**	org.ofbiz.widget.screen.MacroScreenViewHandler	Common handler used to display OFBiz screens.
screenfop	org.ofbiz.widget.screen.ScreenFopViewHandler	Handler that uses XLS-FO formatted templates to generate PDF, PCL, PostScript, etc.
jsp	org.ofbiz.webapp.view.JspViewHandler	Handler for Java Server Pages.

Table 15 – OFBiz view handlers

When a request is made and it needs to execute complex logic before deciding what should be sent to the client an event will be triggered. Events are used to run Java code, a Simple Method or a Groovy[2] script.

The most commonly used event handlers are:

Handler name	Handler Class	Description
Java	org.ofbiz.webapp.event.JavaEventHandler	Static Method Java Event Handler
Soap	org.ofbiz.webapp.event.SOAPEventHandler	SOAP event handler.
xmlrpc	org.ofbiz.webapp.event.XmlRpcEventHandler	XML RPC event handler.
Service	org.ofbiz.webapp.event.ServiceEventHandler	Handler used to invoke OFBIz services.
service-multi	org.ofbiz.webapp.event.ServiceMultiEventHandler	Event handler for running a service multiple times; for bulk forms
service-stream	org.ofbiz.webapp.event.ServiceStreamHandler	Handler for services that need to access raw streams when receiving requests from the clients. The input and output streams are the only parameters.
Simple	org.ofbiz.webapp.event.SimpleEventHandler	Simple event handler. Used to invoke events defined using Mini-Language.
Groovy	org.ofbiz.webapp.event.GroovyEventHandler	Groovy event handler. Used to

[2] http://www.groovy-lang.org/

		invoke events defined using Groovy.
Rome	org.ofbiz.webapp.event.RomeEventHandler	Rome event handler. Used to invoke events defined using Rome e.g. RSS feeds.

Table 16 – OFBiz event handlers

3.2.1.2 Request Maps

Request maps are responsible for specifying the actual logic to be executed when a request is made to a specific URL. The following XML is the definition for the action responsible for creating a new credit card.

```
<request-map uri="createCreditCard">
  <security https="true" auth="true" />
  <event type="simple"
      path="component://accounting/script/org/ofbiz/accounting/
      payment/PaymentMethodEvents.xml"
      invoke="createCreditCard" />
  <response name="success" type="view" value="viewprofile" />
  <response name="address" type="view" value="editcontactmech" />
  <response name="error" type="view" value="editcreditcard" />
</request-map>
```

To create a new credit card the user must send a request to the application module, in this case it is the party module, with the "createCreditCard" in the path, as follows:

Module	Required for URL pattern	Request
/party	/control	/createCreditCard

Table 17 – OFBiz request URL composition

Indirect Requests

Indirect requests are request-maps that cannot be directly invoked by the end-users, meaning that if the user fired a request to an action that does not allow direct requests the user would receive the "Unknown request" error. Indirect requests are useful when the need to share some request map logic between requests maps exists, e.g. the "calcTax" request map used to calculate the tax when buying something does not allow direct requests and it is invoked after a successful calculation of the shipping fee or upon a quick check out request.

```
<request-map uri="calcTax">
  <security direct-request="false" />
  <event type="java"
      path="org.ofbiz.order.shoppingcart.CheckOutEvents"
    invoke="calcTax" />
  <response name="success" type="view" value="confirm" />
  <response name="error" type="request" value="orderentry" />
</request-map>
```

```
<request-map uri="calcShipping">
  <security direct-request="false" />
  <event type="java"
    path="org.ofbiz.order.shoppingcart.shipping.ShippingEvents"
    invoke="getShipEstimate" />
  <response name="success" type="request" value="calcTax" />
  <response name="error" type="request" value="orderentry" />
</request-map>
```

To disable direct requests the value of the “direct-request” attribute of the “security” element should be set to false in the request map definition.

Events

Events contain the business logic of the request, like fetching records from the database for display, reading the user inputs and updating a specific record, etc. Usually the events will return a string or set a context value that will be used by OFBIz to determine the appropriate response. Some of the simplest business logic, like CRUD operations on the database can actually be specified on the view definition and do not require an event, as we will see later. Events are used when:

- The code needs to be executed independently from the response;
- The event will decide what response needs to be sent to the client;

Java Events

Java events are java methods that must obey to the following rules:

- must be a static method;
- must return a java.lang.String;
- must receive two parameters:
    - javax.servlet.http.HttpServletRequest;
    - javax.servlet.http.HttpServletResponse.

The HttpServletRequest parameter can be used to access data from the users request such as URL parameters, HTTP headers, cookies, among other things. The most common usage is to retrieve URL/post parameters which can be done using the “getParameter” method:

```
String productId = request.getParameter("PRODUCT_ID");
String productIdTo = request.getParameter("PRODUCT_ID_TO");
String productAssocTypeId =
request.getParameter("PRODUCT_ASSOC_TYPE_ID");
String fromDateStr = request.getParameter("FROM_DATE");
```

The return of a Java event is used to choose an appropriate response for the request. The returned String must match one of the values in the “name” attribute of a “response” element within the request map. Example:

```
public static Map callingServiceTwo(DispatchContext dctx, Map
context) {
  String userId = (String) context.get("userId");
  Map resultMap = null;
  if (userId.equals("10000")) {
    resultMap = ServiceUtil.returnSuccess("Welcome John!");
  } else {
    resultMap = ServiceUtil.returnError(
      "Welcome unknown user");
  }
  return resultMap;
}
```

The ServiceUtil class provides methods to return “success” or “error” values with messages. Alternatively, a string with the response can be returned.

If no event is present in the request map the default “success” response will be chosen, however, if an event is present but it returns a string that does not match any of the defined response names then no response will be provided (the end-user will see a blank page on the browser).

3.2.1.3 Responses

The response elements dictate what will be done with the result of the processing of the request. There are three types of possible responses:

- View response;
- Request response;
- Request-redirect response;
- Request-redirect-noparam response;

The view response will render an OFBiz view and send it as response.

```
<request-map uri="setCustomer">
  <security https="true" auth="true"/>
  <response name="success" type="view" value="custsetting"/>
</request-map>
```

The “request” response is used to fire a request to another action, leaving the response to the client up to the logic of the requested action.

```
<request-map uri="createCustomer">
  <security https="true" auth="true" />
  <event type="simple"
    path="component://order/script/org/ofbiz/order/
      customer/CustomerEvents.xml"
    invoke="createCustomer" />
  <response name="success" type="request" value="finalizeOrder" />
  <response name="error" type="view" value="custsetting" />
</request-map>
```

The “request-redirect” will send a redirect response to the end-user browser, which will cause it to fire a request to a specific URL.

```
<request-map uri="makeQuickReturn">
  <security https="true" auth="true" />
  <event type="service-multi"
      invoke="createReturnAndItemOrAdjustment" />
  <response name="success" type="request-redirect"
      value="returnItems">
    <redirect-parameter name="returnId" />
  </response>
  <response name="error" type="view" value="orderview" />
</request-map>
```

The request-redirect-noparam response is similar to the request-redirect response but the new request won't have any of the parameters sent in the first request.

3.2.1.4 View Maps

As shown above, one of the possible return types for a request is a view. Whenever an action specifies a return of type "view" the value specified should be the name of an existing view map.

View maps are declared using the view-map element. The view map element can have the following attributes:

Attribute name	Required?	Description
name	Yes	Name of the view map.
content-type	No	HTML content-type.
encoding	No	Charset in the HTML sense. By default "text/html" is used. If the encoding is "none" then no charset will be used.
info	No	Extended information passed to the view handler.
no-cache	No	Send no-cache headers if set to true.
page	No	The page mapped to this view.
Type	No	The name of the view handler that will render the output: screen, screenfop, ftl etc... A most comprehensive list can be found in the common-controller.

Table 18 – View-map element attributes

The following XML shows an example of how to use the view map element.

```
<request-map uri="main">
  <security https="true" auth="true" />
  <response name="success" type="view" value="main" />
</request-map>
<view-map name="main" type="screen"
 page="component://reports/widget/reports/CommonScreens.xml#main"/>
```

3.2.2 Authentication

If we are developing a new module for the OFBiz backoffice we need to make sure that the user is logged in and if not, redirect the request to the OFBiz login page. The controller specification has two main tags:

- Preprocessor – logic that must be executed before any request to the module
- Postprocessor – logic that must be executed after any request to the module

To allow access to the module only to authenticated users, a new event must be fired before all requests. This event is a common OFBiz event that is used in all application modules of the OFBiz back-office and it would be registered as follows:

```
<preprocessor>
  <!-- Events to run on every request before security
       (chains exempt) -->
  <event name="check509CertLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="check509CertLogin" />
  <event name="checkRequestHeaderLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkRequestHeaderLogin" />
  <event name="checkServletRequestRemoteUserLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkServletRequestRemoteUserLogin" />
  <event name="checkExternalLoginKey" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="checkExternalLoginKey" />
  <event name="checkProtectedView" type="java"
    path="org.ofbiz.webapp.control.ProtectViewWorker"
    invoke="checkProtectedView" />
  <event name="extensionConnectLogin" type="java"
    path="org.ofbiz.webapp.control.LoginWorker"
    invoke="extensionConnectLogin" />
</preprocessor>
```

These preprocessor events are required for all controllers and as such are defined in the /framework/common/webcommon/WEB-INF/common-controller.xml file.

When changing from one module to another, an external login key must be specified in the URL parameters for the user to stay logged in.

```
https://127.0.0.1:8443/humanres/control/main?externalLoginKey=EL460517773307
```

The above preprocessor is intended to be used to share a login key between application modules but it does not automatically restrict access to authenticated users.

To allow access only to authenticated users to any given request, the “auth” attribute of the “security” element must be specified in the request definition with a value of “true”.

The “auth” attribute works by firing a request to the “checkLogin” action of the common OFBiz control servlet. This action will validate if the user is logged in and if not, the user will be redirected to the login page.

When dealing with user credentials and other sensitive data in web applications the HTTPS protocol should be used to ensure proper security over the communication between the end-user and the server. OFBiz can work with both HTTP and HTTPS, which is opt-in. To require a secure connection to fire a given request the “https” attribute of the “security” element must be specified in the request-map definition, as shown below:

```
<request-map uri="viewprofile">
  <security https="true" auth="true" />
  <response name="success" type="view" value="viewprofile"
    save-home-view="true" />
</request-map>
```

3.2.3 Authorization

Authentication guarantees that the user must have an account in the system and use their credentials to be able to access OFBiz. This however might not be enough to decide if the user can view or change something, if only authorization was used all users would be able to create or remove other users. A common practice to overcome this problem is to associate users with Roles (e.g. Administrator, Content manager, Recruiter, etc.) or assign users with specific permissions. OFBiz has the concept of Roles and also has the concept of Security Groups that act as a group of permissions and user logins are then associated with security groups. In the end, developers are capable of restricing the access to a given content based on the following:

- The role of the user;
- Whether or not the user has a given permission;

Permissions are identifier by a name composed of two components, a permission and an action. The permission usually refers to an entity or module while the action refers to the action that is being performed to that entity or within that module.

```
<section>
    <condition>
        <if-has-permission permission="CATALOG" action="_VIEW"/>
    </condition>
    <widgets>
...
```

The two components might also appear together especially in Java. User permissions can be verified in Java by invoking the “hasPermission” method of an instance of “org.ofbiz.security.Security”. The following event illustrates how permissions can be checked in a Java event.

```
Security security = (Security)request.getAttribute("security");
if (security.hasPermission("CATALOG _VIEW", request.getSession()))
{
  request.setAttribute("_EVENT_MESSAGE_", "You have access!");
}
else {
  request.setAttribute("_EVENT_MESSAGE_", "You DO NOT have
  access! You are denied!");
}
  return "success";
}
```

When declaring a service we can also specify what permissions are required in order to be able to execute the service. This can be done using the “required-permissions” and “check-permission” elements. The “required-permission” element is used to combine multiple checks. The following example would allow the user to execute the service if the had the “CATALOG_UPDATE” or the “CATALOG_ADMIN” permission.

```
<required-permissions join-type="OR">
  <check-permission permission="CATALOG_UPDATE"/>
  <check-permission permission="CATALOG_ADMIN"/>
</required-permissions>
```

To validate ta single permission the “required-permission” element is not necessary. Alternatively we can check if the user has a given role type to allow the execution of the service.

```
<check-role-member role-type="CUSTOMER"/>
```

If the permission check is complex and involves checking for example the values of an entity (e.g. check if the current user was the one that created the entity) we have to use a permission service instead.

```
<service name="cancelPartyInvitation" engine="simple"
      location="component://party/script/org/ofbiz/party/party/
      PartyInvitationServices.xml" invoke="cancelPartyInvitation">
  <permission-service
      service-name="cancelPartyInvitationPermissionCheck"/>
  <attribute name="partyInvitationId" type="String"
      mode="IN" optional="false"/>
</service>
```

The service shown above relies on the “cancelPartyInvitationPermissionCheck” permission service check if the user is authorized to execute the service.

```
<service name="cancelPartyInvitationPermissionCheck"
      engine="simple"
      location="component://party/script/org/ofbiz/party
            /party/PartyPermissionServices.xml"
      invoke="cancelPartyInvitationPermissionCheck">
  <description>
    Performs cancel PartyInvitation security check.
      The userLogin partyId must equal the partyId/partyIdFrom in
      PartyInvitation OR partyId fetched using emailAdress in
      PartyInvitation.
      The user with PARTYMGR_UPDATE permission can
      also perform this function.
  </description>
  <implements service="permissionInterface"/>
  <attribute name="partyInvitationId" type="String"
      mode="IN" optional="false"/>
</service>
```

```
<!-- Service Engine Interfaces -->
<service name="permissionInterface" engine="interface">
  <description>Interface to describe base parameters
      for Permission Services</description>
  <attribute name="mainAction" type="String"
      mode="IN" optional="true">
    <description>The action requiring permission. Must be one
      of ADMIN, CREATE, UPDATE, DELETE, VIEW.</description>
  </attribute>
  <attribute name="primaryPermission" type="String"
      mode="IN" optional="true">
    <description>The permission to check - typically the name
      of an application or entity.</description>
  </attribute>
  <attribute name="altPermission" type="String"
      mode="IN" optional="true">
    <description>Optional alternate permission to check.
      If the primary permission check fails, the alternate
      permission will be checked.</description>
  </attribute>
  <attribute name="resourceDescription" type="String"
      mode="IN" optional="true">
    <description>The name of the resource being accessed
      - defaults to service name.</description>
  </attribute>
  <attribute name="hasPermission" type="Boolean"
      mode="OUT" optional="false">
    <description>Contains true if the requested permission
      has been granted.</description>
  </attribute>
  <attribute name="failMessage" type="String"
      mode="OUT" optional="true">
    <description>Contains an explanation if the permission
      was denied.</description>
  </attribute>
</service>
```

```
<simple-method method-name="cancelPartyInvitationPermissionCheck"
short-description="Cancel PartyInvitation Permission Logic">
  <set field="hasPermission" type="Boolean" value="false"/>
  <if-has-permission permission="PARTYMGR_UPDATE" action="_UPDATE">
    <set field="hasPermission" type="Boolean" value="true"/>
    <field-to-result field="hasPermission"/>
  </if-has-permission>
  ...
  <if-compare field="hasPermission"
      operator="not-equals" value="true">
    <property-to-field
      property="PartyInvitationCancelPermissionError"
      field="failMessage" resource="PartyUiLabels"/>
    <field-to-result field="hasPermission"/>
    <field-to-result field="failMessage"/>
  </if-compare>
</simple-method>
```

To validate authorization in Simple Services or Simple Events we can used the same “check-permission” element. The fail property would represent the fail message to be returned if the the user didn’t have the specified permission.

```
<check-permission permission="PARTYMGR" action="_UPDATE">
  <fail-property resource="PartyUiLabels"
      property="PartyPermissionErrorForThisParty"/>
</check-permission>
```

3.3 Screen Widgets

The “View” part of the MVC implementation in OFBiz is composed of Screen Widgets. Every view is a screen widget or is contained within one. Views are responsible to present information to the end-user as well as providing the means to change it.

The screen widgets are defined per application module thus they are contained within an XML file within the “widgets” folder of a given OFBiz module. To be able to use screen widgets the controller must be aware of what class is responsible for handling the views, the default handler is the “org.ofbiz.widget.screen.MacroScreenViewHandler”.

```
<handler name="screen" type="view"
  class="org.ofbiz.widget.screen.MacroScreenViewHandler" />
```

A screen is defined by using the “screen” XML element and a given name. Within the screen element a section needs to be defined (one section per screen). Sections act as containers for the view elements and can contain logic to decide whether or not show a given element. This can be accomplished by using the If-then-else structure of the section element. The implementation of conditional screens can be achieved by using the “condition” element, which represents the “if” part of the if-then-else structure.

The following conditions are available in OFBiz:

- and
- or
- xor
- not
- if-service-permission
- if-has-permission
- if-entity-permission
- if-validate-method
- if-compare
- if-compare-field
- if-regexp
- if-empty
- if-empty-section

If the condition evaluates to true the “then” part will be executed, as expected. In the screen widgets, the “then” part is composed by two elements: the “actions” and the “widgets”. When the condition is evaluated to false, only the “fail-widgets” element is executed, meaning that the content within the “actions” and “widgets” elements will be skipped.

The “actions” element is where the logic to retrieve elements from the database, manipulation or transformation of input parameters should be contained. When communicating with the database only “read” operations are allowed here. The widget actions can be categorized into three distinct groups:

- The actions that retrieve values from the database and place it in variables;
- The actions that invoke scripts or services;
- The actions that manipulate variables.

3.3.1 Database retrieval actions

- entity-and – Used to find an entity by specifying a set of filters that will be concatenated by an “and” operator;
- entity-condition – The entity condition element is used to find entities by some specific criteria;
- entity-one – The entity-one element is used to find an entity by its primary-key;
- get-related-one – The get-related-one element is used to retrieve an entity that is related to another entity based on the declared relations of the primary entity;
- get-related – The get-related element is similar to the get-related-one as it is also based on the declared relations of an entity but instead of retrieving only one value it is used to retrieve a list of related entities (e.g. entities that have N records related to 1 entity).

3.3.2 Scripts and service invocation actions

- script – used to invoke a script (e.g. a Groovy script);
- call-service – used to invoke an OFBiz service.

3.3.3 Variable manipulation actions

- property-map – The property-map element provides access to OFBiz text resources by mapping all properties of a given resource to a specified map variable;
- property-to-field – The property-to-field element is used to retrieve the value of a single property from an OFBiz resource;
- set – used to declare a variable or set the value of an existing variable.

A detailed description of these elements can be found in the Mini-Language chapter.

The widgets element acts as a container for the elements that will be displayed, like images, labels, links, buttons, etc.

3.3.4 Context Variables

Screen widgets have access to a pre-defined set of OFBiz objects. These objects can be used to access HTTP request parameters, obtain information regarding the user login, etc. The list of available context variables is listed in the Mini-Language chapter.

3.3.5 Defining a screen widget

Screen widgets define what HTML will be written to the response for a given request. That said, the controller must have a request map that returns a view. Views are placed within an XML file whose name should end with “Screens” (e.g. PartyScreens, OrderScreens, etc.) and should be placed within the widget folder of the given module (e.g. applications/party/widget /partymgr/PartyScreens.xml). Assume that the PartyScreens.xml has a view called “findparty”, this view could be referenced in a request map by adding “#<name of the view>” after the location of the file where it is located, e.g.

```
<view-map name="main" type="screen"
  page="component://party/widget/partymgr/
      PartyScreens.xml#findparty"/>
```

As said above, screen widgets are declared using the screen element that must contain at least one section element and one widgets element within the section.

The first section element is required for a screen element and can be understood as the body for the page that will be rendered. There must be exactly one section per screen element.

The widgets element is where the content of the screen is placed. The widgets element supports multiple elements to render images, labels, links, etc.

The following XML illustrates the simplest possible OFBiz screen widget, containing only one label.

```
<screen name="SimplestScreen">
  <section>
    <widgets>
      <label text="Simplest Screen possible in OFBiz!" />
    </widgets>
  </section>
</screen>
```

A screen section can have conditions and actions, used for example to show or hide some content based on the value of a variable or request parameter. These can be achieved using the “conditions” and “actions” element.

The conditions element acts as an “If” and if the result of the conditions element evaluates to true the then part of the screen, composed of the actions and widgets elements, is executed otherwise the else part, composed only of the “fail-widgets” element is executed.

The next XML shows a screen that receives a request parameter called “show” and displays different labels based on the value of that parameter.

```
<screen name="ConditionalScreen">
  <section>
    <condition>
      <if-compare field="parameters.show" operator="equals"
        value="all" />
    </condition>
    <actions>
      <set field="showing" value="all" />
    </actions>
    <widgets>
      <label
        text="Condition passed. Showing widgets element.
            Showing: ${showing}" />
    </widgets>
    <fail-widgets>
      <label
        text="Condition failed! Showing fail-widgets element.
            Showing: is: ${showing}" />
    </fail-widgets>
  </section>
</screen>
```

The actions element is commonly used to execute CRUD operations or invoke services.

In the example above, if the value of the request parameter “show” was not “all” then the showing value would not be set, because the actions were not executed. To solve this we another section element could be used within the fail-widgets element:

```
<screen name="ConditionalScreen">
  <section>
    <condition>
      <if-compare field="parameters.show" operator="equals"
        value="all" />
    </condition>
    <actions>
      <set field="showing" value="all" />
    </actions>
    <widgets>
      <label
        text="Condition passed. Showing widgets element.
            Showing: ${showing}" />
    </widgets>
    <fail-widgets>
      <section>
        <actions>
          <set field="showing" value="none" />
        </actions>
        <widgets>
          <label
            text="Condition failed! Showing fail-widgets element.
                  Showing: is: ${showing}" />
        </widgets>
      </section>
    </fail-widgets>
  </section>
</screen>
```

Using a section element creates a context, variables declared within an inner section, such as the one within the fails widget would not be available to the outside section while the opposite is true, inner sections inherit the context of the parent section.

Screens can be reused within other screens by using the include-screen element, as shown below:

```
<screen name="NestedScreen">
  <section>
    <widgets>
      <label text="This is the simple screen content:" />
        <include-screen name="SimplestScreen" />
    </widgets>
  </section>
</screen>
```

We have seen that screens have a global context, where a predefined set of variables can be accessed. When we include another screen we are “creating” a new variable context that inherits all of the variables that exist in the parent screen. All of the variables declared within the nested screen will only be visible to the parent screen if explicitly marked as global (by using the global attribute of the set element). Variables that are not global will only be declared within the context of the screen. Consider the following example:

```
<screen name="ReportSaft">
  <section>
    <actions>
      <set field="global1" value="Global 1" global="true" />
      <set field="local1" value="Local 1" global="false" />
    </actions>
    <widgets>
      <include-screen name="NestedContext"/>
      <container></container>
      <container>
        <label text="MAIN CONTEXT:" />
      </container>
      <container>
        <label text="global1 is ${global1}"/>
      </container>
      <container>
        <label text="global2 is ${global2}"/>
      </container>
      <container>
        <label text="local1 is ${local1}"/>
      </container>
      <container>
        <label text="local2 is ${local2}"/>
      </container>
    </widgets>
  </section>
</screen>

<screen name="NestedContext">
  <section>
    <actions>
      <set field="global2" value="Global 2" global="true" />
      <set field="local2" value="Local 2" global="false" />
    </actions>
    <widgets>
      <container>
        <label text="NESTED CONTEXT:" />
      </container>
      <container>
        <label text="global1 is ${global1}"/>
      </container>
      <container>
        <label text="global2 is ${global2}"/>
      </container>
      <container>
        <label text="local1 is ${local1}"/>
      </container>
      <container>
        <label text="local2 is ${local2}"/>
      </container>
    </widgets>
  </section>
</screen>
```

This example would print the following output:

NESTED CONTEXT:
global1 is Global 1
global2 is Global 2
local1 is Local 1
local2 is Local 2
MAIN CONTEXT:
global1 is Global 1
global2 is Global 2
local1 is Local 1
local2 is

3.3.6 Screen decorators

OFBiz uses the screen decorator pattern to render most of its screens. This helps to centralize the logic to render the application header and footer in one place and leave the actual content to be declared in each screen definition. The following XML shows the definition for the header and footer screens.

```
<screen name="header">
  <section>
    <widgets>
      <label text="This is the header"></label>
    </widgets>
  </section>
</screen>
<screen name="footer">
  <section>
    <widgets>
      <label text="This is the footer"></label>
    </widgets>
  </section>
</screen>
```

To use these screens in the simplest screen shown above we could simply include the screens but this is not the best approach. What is done in OFBiz instead is that a screen decorator will be created and that screen decorator will have one or many placeholders that the screens can customize, in this case only one will be used and it will be named “body”. This placeholder is defined using the decorator-section-include element and providing a name.

```
<screen name="simple-decorator">
  <section>
    <widgets>
      <include-screen name="header" />
      <decorator-section-include name="body" />
      <include-screen name="footer" />
    </widgets>
  </section>
</screen>
```

The following screen will use the new decorator (using the decorator-screen element) and specify the content for the “body” section (using the decorator-section element with the correct name) with the contents of the simplest screen show above.

```
<screen name="CompoundedScreen">
  <section>
    <widgets>
      <decorator-screen name="simple-decorator">
        <decorator-section name="body">
          <include-screen name="SimplestScreen" />
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

OFBiz already provide a decorator that is used by every module but it is not ready to use, it has some specifications that vary by module, e.g. the application titles and menus.

Assuming that a new OFBiz module is created and that it is called “Reports” the following XML shows the basic decorator that needed to be declared and that use the decorator that is provided by OFBiz.

```
<screen name="main-decorator">
  <section>
    <actions>
      <property-map resource="CommonUiLabels" map-name="uiLabelMap"
        global="true" />
      <property-map resource="ReportsUiLabels"
        map-name="uiLabelMap"
        global="true" />
      <set field="layoutSettings.companyName"
        from-field="uiLabelMap.ReportsCompanyName"
        global="true" />
      <set field="layoutSettings.companySubtitle"
        from-field="uiLabelMap.ReportsCompanySubtitle"
        global="true" />
      <set field="applicationMenuName" value="ReportsAppBar"
        global="true" />
      <set field="applicationMenuLocation"
        value="component://reports/widget/reports/ReportsMenus.xml"
        global="true" />
      <set field="applicationTitle"
        value="${uiLabelMap.ReportsApplication}"
        global="true" />
    </actions>
    <widgets>
      <include-screen name="ApplicationDecorator"
        location="component://commonext/widget/
            CommonScreens.xml" />
    </widgets>
  </section>
</screen>
```

A simple screen using this decorator could be declared as follows.

```
<screen name="ReportSaft">
  <section>
    <widgets>
      <decorator-screen name="main-decorator"
        location="${parameters.mainDecoratorLocation}">
        <decorator-section name="body">
          <label text="Content…"></label>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

3.3.7 Menus

OFBiz back-end pages display a menu on the top that changes depending on the page that is currently being displayed. These menus must be defined in an XML file named "<module>Menus.xml" and that is placed in the same folder as the module screens xml file.

Menus are declared using the root Menus element and each menu is declared using the Menu element. OFBiz menus should extend the "CommonAppBarMenu" menu and the "component://common/widget/CommonMenus.xml" resource.

Menu items can be composed of other menus or hyperlinks, using the link element as shown below:

```
<menus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/
      widget-menu.xsd">
  <menu name="ReportsAppBar" title="${uiLabelMap.Reports}"
    extends="CommonAppBarMenu"
    extends-resource="component://common/widget/CommonMenus.xml">
    <menu-item name="reportsaft" title="${uiLabelMap.ReportSaft}">
      <link target="ReportSaft" />
    </menu-item>
  </menu>
</menus>
```

This menu is then displayed on the page by setting the applicationMenuName and applicationMenuLocation fields of the common ofbiz decorator as shown in the simple screen decorator above.

The first menu level, the one that contains one entry for each module, is automatically selected with the menu declaration in the common screen decorator for the module, as shown below:

```
<screen name="main-decorator">
  <section>
    <actions>
      ...
      <set field="applicationMenuName" value="ReportsAppBar"
        global="true" />
      <set field="applicationMenuLocation"
        value="component://reports/widget/reports/ReportsMenus.xml"
        global="true" />
      ...
    </actions>
    <widgets>
      ...
    </widgets>
  </section>
</screen>
```

The value for the “applicationMenuName” variable should be the name of the menu and the “applicationMenuLocation” should contain the location of the file where the menu with the specified name is located.

For the second menu level, the one that contains the specific menu entries for each module and varies between modules, the active menu entry is selected by assigning a value to the “headerItem” variable. Note that the menu specified above contains one item name “reportSaft”, this is the name that we need to assign to the variable to indicate the active menu entry.

```
<screen name="ReportPending">
  <section>
    <actions>
      ...
      <set field="headerItem" value="reportsaft" />
    </actions>
    <widgets>
      ...
    </widgets>
  </section>
</screen>
```

3.3.8 Forms

One of the required element of any application is the existence of forms, and OFBiz is no different. Forms are supported by declaring them in XML. Each module has its own forms file that should be named “<Module>Forms.xml” and placed in the widget folder of that module.

Forms are specified using the Form element. The form element has the following attributes

- Name – specifies the name of the form;
- Type – specifies the type of the form. Can have the following values:
  - Single
  - List
  - Multi
  - Upload
- Target – name of the action that will receive the post request with the form values.

The form element can have several field elements, one for each input of the form. Each field also has a sub-element that specifies the type of the input field, e.g. text, dropdown, date, etc.

The following xml shows a simple form:

```
<forms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/
      widget-form.xsd">
  <form name="ReportSaftGenerationForm"
    type="single" target="GenerateSaft">
    <field name="dateBegin">
      <date-time />
    </field>
    <field name="dateEnd">
      <date-time />
    </field>
    <field name="submit">
      <submit />
    </field>
  </form>
</forms>
```

Displaying fields with multiple values, such as dropdowns required a bit more work.

```
<field name="timePeriod" title="${uiLabelMap.SaftTimePeriodTitle}">
  <drop-down allow-empty="false" allow-multiple="false">
    <entity-options entity-name="CustomTimePeriod"
      key-field-name="customTimePeriodId"
      description="${periodName}" filter-by-date="false">
      <entity-constraint name="periodTypeId"
            operator="equals" value="FISCAL_YEAR"/>
      <entity-order-by field-name="periodName"/>
    </entity-options>
  </drop-down>
</field>
```

The XML above would display a dropdown with all the “CustomTimePeriod” entity values and for each one it will render an HTML option element with the key being the value of the “customTimePeriodId” property and the text being the “periodName” property of the same entity. If the “filter-by-date” attribute is set to true only the entities that are valid at the current date will be displayed. It is also possible to apply constraints and order to the entities, in the case above we are displaying only entites that have a value of “FISCAL_YEAR” for the “periodTypeId” property. The entities will also be sorted by the “periodDate” property ascendently.

3.3.9 FreeMarker

FreeMarker is a template engine. It is a Java library used to generate HTML based on a template and on a model.

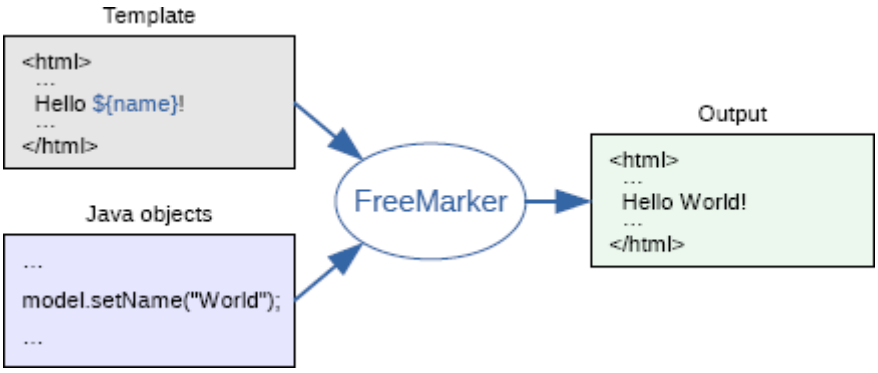


Figure 24 – FreeMarker overview

FreeMarker templates should be used to handle only how the data is displayed, querying the database and preparing data to be displayed should be done in OFBiz events, services, scripts or screens. Using FreeMarker templates requires users to use the "platform-specific" element within a screen since the FreeMarker template used will allways generate HTML.

```
<platform-specific>
  <html>
    <html-template
      location="component://reports/webapp/reports/
            reports/ReportStatus.ftl" />
  </html>
</platform-specific>
```

Within the FreeMarker template developers have access to the variables that are in the context of the screen location where the platform-specific element is used.

The following code shows how to display a table with the user identifier and user name for all the users of a list, if the list has any users.

```
<#if usersList?has_content>
  <table class="basic-table" cellspacing="0">
    <#-- Header Begins -->
    <tr class="header-row-2">
      <th>${uiLabelMap.UserId}</th>
      <th>${uiLabelMap.UserName}</th>
    </tr>
    <#-- Header Ends-->
    <#assign alt_row = false />
    <#list usersList as user>
      <tr valign="middle"<#if alt_row> class="alternate-row"</#if>>
        <td>${user.userId}</td>
        <td>${user.Name}</td>
      </tr>
    </#list>
  </table>
</#if>
```

FreeMarker templates should be used if none of the other options allow to do what we pretend. They are specially usefull when the screen being developed requires Javascript code.

To be able to execute custom Javascript code in our pages we need to indicate OFBiz that we need a new Javascript file to be served with the screen. This can be done by adding an entry to a global variable that is declared within the common OFBiz screen decorators. The example below will include the "/reports/js/ReportsQueueStatus.js" file. Note that this is the path to the file in the file system.

```
<screen name="ReportPending">
  <section>
    <actions>
      <set field="layoutSettings.javaScripts[]"
           value="/reports/js/ReportsQueueStatus.js"
           global="true" />
      <set field="titleProperty"
           value="PageTitleReportSaftDetails"/>
      <set field="headerItem" value="reportsaft" />
    </actions>
    <widgets>
      <decorator-screen name="main-decorator"
        location="${parameters.mainDecoratorLocation}">
        <decorator-section name="body">
          <include-screen name="ReportResultDisplayInternal" />
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

Let's assume that our script file has a function called "StartReportsStatusUpdateTimer" which is used to periodically invoke an OFBiz action via AJAX and update the user interface accordingly, this function could be called with the following code within the FreeMarker template. In this case we even send the value of a context variable to the javascript.

```
<script type="text/javascript">
   jQuery(document).ready( function() {
        StartReportStatusUpdateTimer('ReportResultPlaceHolder',
            'StatusPlaceHolder', ${reportQueue.reportQueueId});
   });
</script>
```

Our function could have the following code:

```
function getReportStatus(placeHolderId, statusPlaceHolderId,
queueId){
  jQuery.ajax({
    url: 'getReportStatusIdJson',
    async: false,
    type: 'POST', // If changed to GET will result in security
error from OFBiz.
    data: { reportQueueId: queueId },
    success: function(data) {
      jQuery('#' +
statusPlaceHolderId).html(data.reportQueueStatusIdDisplay);

      if (data.reportQueueStatusId !== 'PENDING' &&
            data.reportQueueStatusId !== 'RUNNING' &&
            data.reportQueueStatusId !== 'QUEUED') {
        StopReportStatusUpdateTimer();
        getDownloadDisplay(placeHolderId, queueId);
      }
    }
  });
}
```

This will invoke the “getReportStatusIdJson” that will return a JSON object with status identifier and status name. To do this we must guarantee that our request map invokes a service or a Java event. The service must return exactly what we want to be sent to the client and the java event must add the values to be sent to the client to as attributes of the HTTPRequest. In both cases our responses must all have a type of “request” and a value of “json”. This will cause OFBiz to create an object with the service output attributes, serialize in JSON and send it to the client.

```
<request-map uri="getReportStatusIdJson">
  <security https="true" auth="true" />
  <event type="service" invoke="getReportStatusDisplayForSaft" />
  <response name="success" type="request" value="json" />
  <response name="error" type="request" value="json" />
</request-map>
```

```
<service name="getReportStatusDisplayForSaft"
      engine="java" auth="true"
      invoke="getReportStatusDisplayForSaft"
      location="org.ofbiz.reports.ReportsServices">
  <description>Retrieves the Report Queue Status ID
      of a given report queue entry.</description>
  <attribute name="reportQueueId" mode="IN" type="String"
      optional="false" />
  <attribute name="reportQueueStatusId" mode="OUT" type="String" />
  <attribute name="reportQueueStatusIdDisplay" mode="OUT"
      type="String" />
</service>
```

The following XML excerpt shows the same logic of the XML excerpt above but using a java event instead of invoking a service.

```
<request-map uri="getReportStatusIdJson">
  <security https="true" auth="true" />
  <event type="java" path="org.ofbiz.reports.ReportsEvents"
    invoke="getReportStatusDisplayForSaft" />
  <response name="success" type="request" value="json" />
  <response name="error" type="request" value="json" />
</request-map>
```

```
public static String getReportStatusDisplayForSaft(
      HttpServletRequest request,
      HttpServletResponse response)
      throws GenericEntityException,
            SerializeException,
            FileNotFoundException, IOException {
      Delegator delegator = (Delegator)
            request.getAttribute("delegator");
      Locale locale = UtilHttp.getLocale(request);
      String reportQueueId =
            request.getParameter("reportQueueId").toString();
      String reportQueueStatusId = null;
      String reportQueueStatusIdDisplay = null;

      GenericValue reportQueue = delegator.findOne("ReportQueue",
            false, UtilMisc.toMap("reportQueueId", reportQueueId));

      if(reportQueue == null){
        request.setAttribute(
          "_ERROR_MESSAGE_", "Unable to find job.");
        return "error";
      } else {
            reportQueueStatusId =
                  reportQueue.getString("reportQueueStatusId");
            reportQueueStatusIdDisplay =
                  UtilProperties.getMessage(
                        "ReportsUiLabels", "ReportQueueStatus_" +
                        reportQueueStatusId, locale);
      }

      if(UtilValidate.isEmpty(reportQueueStatusId)){
            request.setAttribute("_ERROR_MESSAGE_",
                  "Unable to find job status id display name.");
        return "error";
      } else {
            request.setAttribute("reportQueueStatusId",
                  reportQueueStatusId);
            request.setAttribute("reportQueueStatusIdDisplay",
                  reportQueueStatusIdDisplay);
            request.setAttribute(ModelService.RESPONSE_MESSAGE,
                  ModelService.RESPOND_SUCCESS);
      }

      return "success";
}
```

4 Service Engine

OFBiz contains a service layer that can be used from the web application, invoked from the outside via HTTP (using SOAP) or scheduled to run in the background at a specified time. OFBiz services can be of many different types: Simple or Java. When using Java, the service will be a static method with the input and output parameters passed in using a Map so they can be serialized and transmitted using SOAP via HTTP, if needed.

Each service must specify the name of the service engine in its definition, this service engine name is used by the Service Dispatcher to choose the appropriate service engine to use. OFBiz has one service dispatcher for each entity delegator. Apart from the language used to code the service itself there must be a definition for the service. This definition indicates OFBiz, among other things, what engine should be used and what parameters the service is expecting. Services are declared using XML via the service configuration file. Since services are identified by its name, defining two services with the same name might cause the first service to be overridden by the second.

4.1 Job Scheduler

OFBiz services can be invoked from the web application, from other services or from the Job Scheduler. The Job Scheduler allows a service to be configured to run with a predefined recurrence e.g. once every hour. The job scheduler is a multi-threaded component that reserves one thread for scheduling/job management and separate threads to run the actual services. This approach prevents long running services from postponing the execution of other services on the queue. System administrators are able to maintain the scheduled jobs by accessing the OFBiz WebTools. The OFBiz Webtools at the following URL:

```
https://127.0.0.1:8443/webtools/control/main
```

By clicking on “Schedule Job” system administrators are capable of scheduling a service to execute periodically or run a preconfigured number of times:

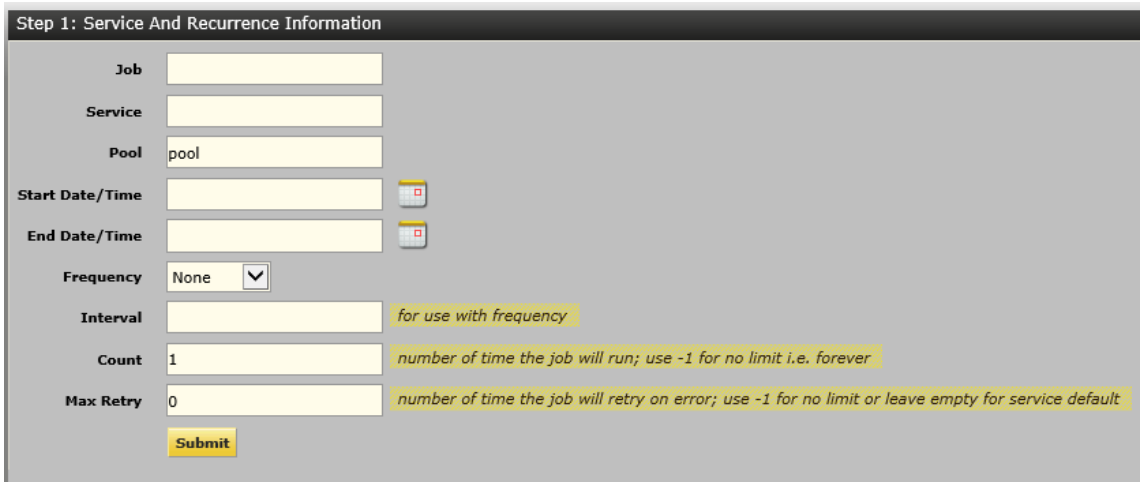


Figure 25 – Scheduling a job

Another way to schedule jobs or simply execute services is via the service details page. This page can be accessed by clicking on a service name on the service reference page.

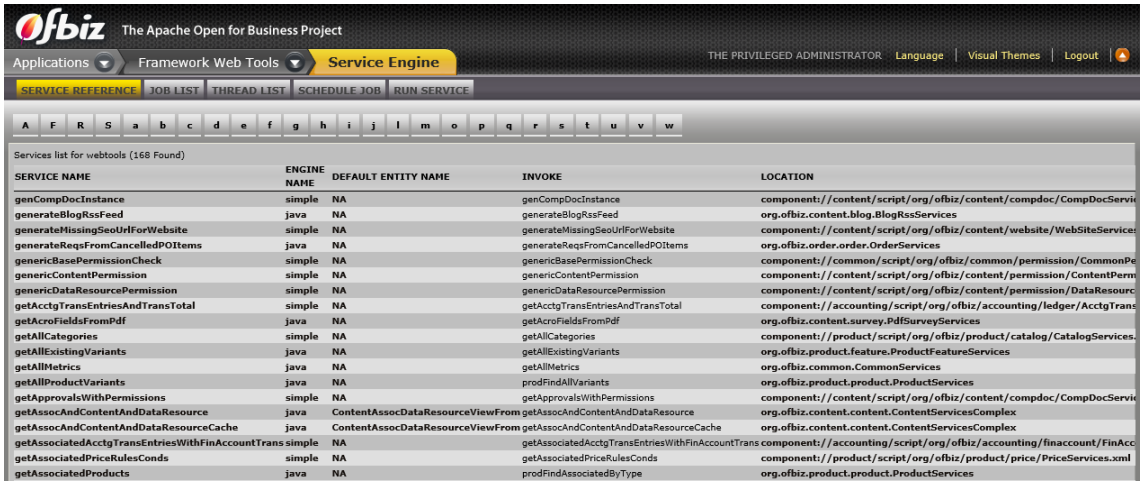


Figure 26 – Service List

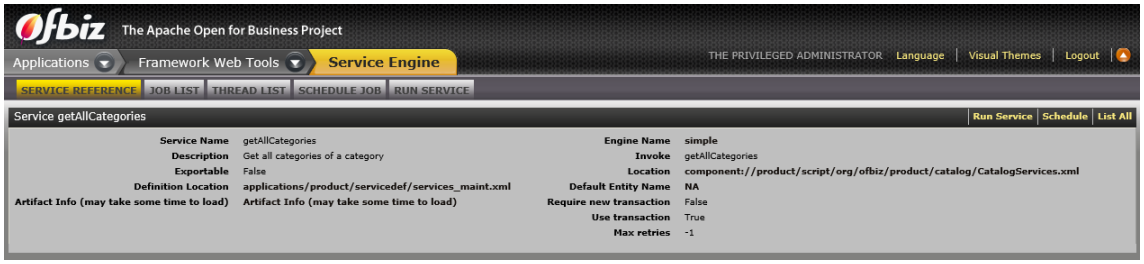


Figure 27 – Scheduled job result

The service detail page allows developers to schedule the job or run the service providing input parameters and viewing the service outputs.

4.2 Service definition

Services are declared in XML files that are usually named services.xml and placed in the servicedef folder of the OFBiz module. The services.xml file starts with the “services” element and inside each service is declared using the “service” element.

```
<service name="deleteParty" engine="java"
  location="org.ofbiz.party.party.PartyServices"
  invoke="deleteParty"
  auth="true">
  <description>Delete a Party</description>
  <attribute name="partyId" type="String"
    mode="IN" optional="true" />
</service>
```

The following tables will describe the main tags and attributes used for service declarations and then an example declaration will be shown.

Sub-element name	How many?	Description
Description	0 or 1	Simple description for the services declared within the services tag.
Vendor	0 or 1	The vendor of the services.
Version	0 or 1	The version of the services
Created	0 or 1	Creation date of the service.
Service	0 to many	Defines a service.

Table 19 – Sub-elements of the Services element

4.2.1 Service sub-element

Attribute name	Required?	Description
Name	Yes	The name of the service.
Engine	Yes	The type of engine that will execute the service. This property can have values such as “simple” for simple method services and “java” for java services.
Location	No	The location of the file where the service logic is defined.
Invoke	No	The file specified on the “Location” attribute can specify multiple services. The invoke attribute is used to specify the “method” that will be invoked.
Auth	No	True if the service requires the user to be authenticated, false otherwise.
Export	No	
Validate	No	
Default-entity-name	No	The name of the default entity that the service relies on.
Use-transaction	No	If set to true and there is no transaction already in place the Service Engine will begin one. If set to false or there is a transaction already in place the Service Engine will do nothing (this also means that if set to false and a transaction is already in place it will do nothing).
Require-new-transaction	No	If set to true and there is a transaction already in place the Service Engine will suspend that transaction, begin a new one just for this service, commit or rollback the local transaction when the service is complete, and will resume the original transaction. If set to true and there is no transaction already in place it will just begin a transaction and manage it as would be done for a normal user-transaction=true. If use-transaction=false this setting is ignored. Beware: using require-new-transaction=true in a service called (maybe not directly) by a pre-invoke or earlier event (preprocessor, firstvisit and so on) is not yet supported

hideResultInLog	No	If set to true the result will be hidden from possible exposition in LocalDispatcher.runSync()
Transaction-timeout	No	Defines the timeout for the transaction, in seconds. Defaults to the value set in the TransactionFactory being used (typically 60 seconds). This value is only used if this service begins a transaction (either require-new-transaction=true, or use-transaction=true and there is no other transaction already in place). If use-transaction=false this setting is ignored.
Max-retry	No	The maximum number of retries that will be done if the service execution fails.
Debug	No	If set to true will enable verbose debugging when the service is called.
Semaphore	No	Defines how concurrent calls to this service should be handled: none: multiple calls to this service may run concurrently wait: while this service is running, queue any subsequent calls fail: while this service is running, fail any subsequent calls
Semaphore-wait-seconds	No	When the semaphore attribute is set to “wait” this attribute specifies how many seconds to wait until the service is executed.
Semaphore-sleep	No	When the semaphore attribute is set to “wait” this attribute specifies how often to check if the waiting service can be executed. The value is specified in miliseconds.

Table 20 – Service element attributes

Sub-Element name	How Many?	Description
Description	0 or 1	Description of the service.
Namespace	0 or 1	Namespace of the service
Permission-service	0 or 1	Specifies the name of a service that will be used to check if the service is allowed to be executed.
Required-permissions	0 to many	Alternative to the permission-service tag. Specifies what permissions are required in order for the service to be executed.
Implements	0 to many	Specifies the interface that the service implements.
Metric	0 or 1	Calculate and maintain an average response time for this service. Service metrics can be used for monitoring and reporting. The metric works by gathering statistics until a configurable maximum is reached (number of requests or elapsed time), then the average is calculated. A smoothing factor is used to smooth differences between calculations.

Auto-attributes	0 or 1	When set to true the service will have its attributes defined based on the properties of the default entity (e.g. all entity primary keys and/or foreign keys).
Attribute	0 to many	Specifies a service attribute / parameter.
Override	0 to many	Used to override attributes from the auto-attributes tag.

Table 21 – Sub-elements of the service element

4.2.2 Permission-service sub-element

Attribute name	Required?	Description
Service-name	Yes	The name of the service responsible for executing the authorization validation.
Resource-description	No	This will be used in error messages, et cetera. If not specified will default to a service name.
Main-Action	No	It's an enumeration that can have one of the following values: "CREATE", "UPADTE", "DELETE" and "VIEW".

Table 22 – Permission-service element attributes

4.2.3 Required-permissions sub-element

Attribute name	Required?	Description
Join-type	Yes	Enumeration that can have the following values "AND" or "OR".

Table 23 – Required-permissions element attributes

Sub-element name	How many?	Description
Check-permission	0 to many	Used to specify the name of an application/functionality level security permission to be validated.
Check-role-member	0 to many	Specifies the name of an OFBiz role that the current user must have in order to execute the service.
Permission-service	0 to many	Specifies another service responsible for validating security permissions.

Table 24 – Sub-elements of the Required-permissions element

4.2.4 Check-permission sub-element

Attribute-name	Required?	Description
Permission	Yes	The name of the permission.
Action	No	Used to complement the permission attribute when specifying two part permissions. E.g. the PARTY_UPDATE permission can be specified has "PARTY" permission and "UPDATE" action.

Table 25 – Sub-elements of the Check-permission element

4.2.5 Check-role-member sub-element

This sub-element is deprecated.

4.2.6 Implements sub-element

Attribute-name	Required?	Description
Service	Yes	The name of the service interface.

Optional	No	If set to true all attributes inherited will have be optional whether or not they were in the implemented service definition.

Table 26 – Implements element attributes

4.2.7 Metric sub-element

Attribute-name	Required?	Description
Name	Yes	Each metric must have a unique name.
estimation-size	No	Positive integer number of requests to include in the metrics calculation. Defaults to "100".
estimation-time	No	Positive integer number of milliseconds to include in the metrics calculation. Defaults to "1000".
Smoothing	No	Positive decimal smoothing factor - used to smooth the differences between calculations. A value of "1" disables smoothing. Defaults to "0.7".

Table 27 – Metric element attributes

4.2.8 Auto-attributes sub-element

Sub-element name	How many?	Description
Exclude	0 to many	Specifies what fields should be excluded.

Table 28 – Sub-elements of the Auto-attributes element

Attribute name	Required?	Description
Field-name	Yes	The name of the field that will be excluded.

Table 29 – Auto-attributes element attributes

4.2.9 Attribute sub-element

Sub-element name	How many?	Description
Type-validate	0 to many	Use to configure validations for the service attribute.
Description	0 or 1	Friendly description of the attribute.

Table 30 – Sub-elements of the Attribute element

Attribute name	Required?	Description
Name	Yes	The name of the attribute.
Type	Yes	The type of the attribute.
Mode	Yes	Can have one of the following values “IN”, “OUT” or “INOUT”.
Optional	No	Boolean value.
Default-value	No	The value specified will be used for the attribute if no value is passed in. This will only happen if it is okay to not pass a value in, so if this is set then optional will be set to true. If optional=false and this is set then the value will be overridden and with a value in default-value is will set optional=true anyway.
Form-label	No	The value of the label when displaying the service attributes in a form.
Entity-name	No	When used allows the automatic configuration of attributes based on the specified entity. E.g. the

		attributes of the service can be all entity fields, only the primary key fields or only the non-primary key fields.
Include	No	Used in combination with the entity-name attribute. Specifies what entity fields will be used as service attributes. Can have the following values: all (for all fields), pk (only primary key fields) and nonpk (for non primary key fields)
Request-attribute-name	No	Used to send values from the Http request to the service.
Session-attribute-name	No	Used to send session parameters to the service attributes.
String-map-prefix	No	When specified will combine all attributes that have the specified prefix into a single map. E.g. if the string-map-prefix attribute holds the value of “param_” then all attributes starting with “param_” will be combined into a single map.
String-list-suffix	No	Used to filter attributes using the specified suffix.
Form-display	No	Boolean value.
Allow-html	No	Applies only to String fields. Only checked for incoming parameters/attributes (could change in the future, but this is meant for validating input from users, other systems, etc). Defaults to "none" meaning no HTML is allowed (will result in an error message). If some HTML is desired then use "any". There was previously "safe" but it's deprecated

Table 31 – Attribute element attributes

4.2.10 Override sub-element

Sub-element name	How many	Description
Type-validate	0 to many	Used to validate a given attribute.

Table 32 – Sub-elements of the Override element

Attribute name	Required?	Description
Name	Yes	The name of the attribute.
Type	No	The type of the attribute.
Entity-name	No	Name of the entity attribute to be overridden.
Field-name	No	Name of the field attribute to be overridden.
Mode	No	Can have one of the following values “IN”, “OUT” or “INOUT”.
Optional	No	Boolean value.
Default-value	No	The value specified will be used for the attribute if no value is passed in. This will only happen if it is okay to not pass a value in, so if this is set then optional will be set to true. If optional=false and this is set then the value will be overridden and with a value in default-value is will set optional=true anyway.
Form-label	No	The value of the label when displaying the service attributes in a form.

Form-display	No	Boolean value.
Allow-html	No	Applies only to String fields. Only checked for incoming parameters/attributes (could change in the future, but this is meant for validating input from users, other systems, etc). Defaults to "none" meaning no HTML is allowed (will result in an error message). If some HTML is desired then use "any". There was previously "safe" but it's deprecated

Table 33 – Override element attributes

4.2.11 Fail-message sub-element

Attribute-name	Required?	Description
Message	Yes	The fail message to be displayed.

Table 34 – Fail-message element attributes

4.2.12 Fail-property sub-element

Attribute-name	Required?	Description
Resource	Yes	The name of the text resource that contains the message property.
Property	Yes	The name of the property.

Table 35 – Fail-property element attributes

4.3 Mini-Language Services

OFBiz has many types of different service languages, one of them is the Mini-Language, referred to as Mini-Lang from now on. Services or events created using Mini-Lang are commonly referred to as simple methods. These simple methods have their logic written in XML. The following example shows a simple service that is used to delete a party role.

```
<service name="deletePartyRole" engine="simple"
    location="component://party/script/org/
      ofbiz/party/party/PartyServices.xml"
      invoke="deletePartyRole" auth="true">
  <description>Delete a Party Role (remove a Role from a Party).
    The logged in user must have PARTYMGR_DELETE or have
    permission to change the role of this partyId</description>
  <permission-service service-name="partyRolePermissionCheck"
      main-action="DELETE"/>
  <attribute name="partyId" type="String" mode="IN"
      optional="true"/>
  <attribute name="roleTypeId" type="String" mode="IN"
      optional="false"/>
</service>
```

```
<simple-method method-name="deletePartyRole" short-
      description="Delete a PartyRole">
  <entity-one entity-name="PartyRole" value-field="partyRole"/>
  <remove-value value-field="partyRole"/>
</simple-method>
```

All the elements supported by Mini-Lang are documented in the Mini-Language reference documentation and can be found here:

https://cwiki.apache.org/confluence/display/OFBADMIN/Mini+Language+-+minilang+-+simple-method+-+Reference

4.4 Java Services

Java services are java static methods that are used to code complex business rules that would be otherwise too difficult to maintain using the simple methods written in OFBiz Mini-Lang. These methods receive two parameters:

- the dispatch context, used to retrieve context variables like the datasource delegator;
- a map containing the service parameters.

The Java source-code, used to define services and/or events should be placed within the src folder.

```
public static Map<String, Object> createPerson(
      DispatchContext ctx, Map<String, ? extends Object> context) {

    Map<String, Object> result = FastMap.newInstance();
    Delegator delegator = ctx.getDelegator();
    Locale locale = (Locale) context.get("locale");
    GenericValue userLogin =
      (GenericValue) context.get("userLogin");

    String partyId = (String) context.get("partyId");
    String description = (String) context.get("description");

    //...

    result.put("partyId", partyId);
    result.put(
      ModelService.RESPONSE_MESSAGE,
      ModelService.RESPOND_SUCCESS);
    return result;
}
```

4.4.1 Service attributes

As explained above, the service parameters are defined in the XML file. Service parameters are named service attributes, the following shows the attributes for a service that receives the first name and last name and returns the full name.

```
<attribute name="firstName" type="String" mode="IN"
  optional="true"/>
<attribute name="lastName" type="String" mode="IN"
  optional="true"/>
<attribute name="fullName" type="String" mode="OUT"
  optional="true"/>
```

In the example above all attributes are defined as optional inputs but we could define required attributes by simply changing the value of the “optional” attribute to false. If the required attributes are not provided, the service is not executed and an error message will be displayed.

The input attributes can be accessed in the service by using the Map variable received as a parameter of the method, in this case the parameter is called context. The output attributes should be placed in a Map variable with the expected attribute name. The following example shows the Java code of the service explained above:

```
public static Map handleParameters(
      DispatchContext dctx, Map context) {

      String firstName = (String) context.get("firstName");
      String lastName = (String) context.get("lastName");
      String message = "firstName: " + firstName;
      message = message + "lastName: " + lastName;
      Map resultMap = ServiceUtil.returnSuccess(message);
      resultMap.put("fullName", firstName + " " + lastName);
      return resultMap;
}
```

All services have a few special attributes that are not explicitly declared but they exist. The attributes are:

- userLogin
- locale

The userLogin parameter contains information regarding the authenticated user that is required for permission checks. The locale defines the culture that should be used to retrieve application texts, format dates and currencies.

4.4.2 Service security

By using the context variables shown above services can execute authorization validations and return an error if the user doesn't have the required permissions. Note that these permissions can also be defined in the XML file. Services should only be used for authorization validation when the rules are too complex to define in XML.

```
public static Map serviceWithAuth(
      DispatchContext dctx, Map context) {
  Security security = dctx.getSecurity();
  Map resultMap = null;
  if (context.get("userLogin") == null
    || !security.hasPermission("TEST_VIEW",
      (GenericValue) context.get("userLogin"))) {
      resultMap = ServiceUtil
        .returnError("You have no access!");
  } else {
    resultMap = ServiceUtil.returnSuccess("Welcome!");
  }
  return resultMap;
}
```

4.4.3 Invoking other services

Access to other services is provided by the Local Dispatcher. To invoke a service we simply call the "runSync" method providing the service name and a java Map containing the input attributes.

```
public static Map callingServiceOne(
      DispatchContext dctx, Map context) {

  LocalDispatcher dispatcher = dctx.getDispatcher();
  Map resultMap = null;
  GenericValue userLogin = (GenericValue) context.get("userLogin");
  Locale locale = (Locale) context.get("locale");

  Map serviceTwoCtx = UtilMisc.toMap("firstName",
      "Jorge", "lastName", "Almeida", "userLogin",
      userLogin, "locale", locale);

  try {
    resultMap = dispatcher.runSync("ServiceTwo", serviceTwoCtx);
  } catch (GenericServiceException e) {
    Debug.logError(e, "Learning");
  }

  return resultMap;
}
```

There are three methods that can be used to invoke a service:

- runSync – runs the service synchronously and returns the result as a java Map;
- runSyncIgnore – runs the service synchronously and ignores the result. Nothing is returned;
- runAsync – runs the service asynchronously and nothing is returned.

4.4.4 Implementing interfaces

OFBiz services can implement interfaces. These interfaces are not java interfaces since OFBiz services are static methods and the use of the Java language is not even required. Service interfaces refer to basic service definitions that declare only the service attributes and can be shared among many services. Let's assume that we have to send notifications every time an order is created or is changed and since these notifications have different subject, message and recipient we will create two distinct Java services. Without using interfaces we would have the following service declaration:

```
<service name="sendOrderConfirmation" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderConfirmNotification">
  <description>Send a order confirmation</description>
  <attribute name="orderId" type="String" mode="IN"
    optional="false" />
  <attribute name="comments" type="String" mode="IN"
    optional="true" />
  <attribute name="body" type="String" mode="OUT"
    optional="true" />
  <attribute name="sendTo" type="String" mode="IN"
    optional="true" />
  <attribute name="sendCc" type="String" mode="IN"
    optional="true" />
  <attribute name="orderId" type="String" mode="OUT"
    optional="true" />
  <attribute name="subject" type="String" mode="OUT"
    optional="true" />
  <attribute name="communicationEventId" type="String" mode="OUT"
    optional="true" />
</service>

<service name="sendOrderChangeNotification" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderChangeNotification">
  <description>Send a order notification</description>
  <attribute name="orderId" type="String" mode="IN"
    optional="false" />
  <attribute name="comments" type="String" mode="IN"
    optional="true" />
  <attribute name="body" type="String" mode="OUT"
    optional="true" />
  <attribute name="sendTo" type="String" mode="IN"
    optional="true" />
  <attribute name="sendCc" type="String" mode="IN"
    optional="true" />
  <attribute name="orderId" type="String" mode="OUT"
    optional="true" />
  <attribute name="subject" type="String" mode="OUT"
    optional="true" />
  <attribute name="communicationEventId" type="String" mode="OUT"
    optional="true" />
</service>
```

In this case even though the service attributes are the same they are duplicated. If we create a service interface we are able to remove these duplicated attributes. The service interface will define the attributes and each service will implement the interface.

```
<service name="orderNotificationInterface" engine="interface"
  location="" invoke="">
  <attribute name="orderId" type="String" mode="IN"
    optional="false" />
  <attribute name="comments" type="String" mode="IN"
    optional="true" />
  <attribute name="body" type="String" mode="OUT"
    optional="true" />
  <attribute name="sendTo" type="String" mode="IN"
    optional="true" />
  <attribute name="sendCc" type="String" mode="IN"
    optional="true" />
  <attribute name="orderId" type="String" mode="OUT"
    optional="true" />
  <attribute name="subject" type="String" mode="OUT"
    optional="true" />
  <attribute name="communicationEventId" type="String" mode="OUT"
    optional="true" />
</service>
```

Note the engine attribute set to “interface” in the service definition.

```
<service name="sendOrderConfirmation" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderConfirmNotification">
  <description>Send a order confirmation</description>
  <implements service="orderNotificationInterface" />
</service>

<service name="sendOrderChangeNotification" engine="java"
  require-new-transaction="true" max-retry="3"
  location="org.ofbiz.order.order.OrderServices"
  invoke="sendOrderChangeNotification">
  <description>Send a order notification</description>
  <implements service="orderNotificationInterface" />
</service>
```

4.5 Service Groups

Service groups are used to simplify the invocation of multiple services that must be executed in a specific order. Service groups define the set of services that are executed and the name of the service group. Service Groups can be invoked in the same way as a normal service. If the result-to-context attribute is set to true, the result of the service will be placed in the context for the other services to use.

```
<group name="createCreditCardAndAddress" send-mode="all">
  <invoke name="createPostalAddress" mode="sync"
    result-to-context="true" />
  <invoke name="createCreditCard" mode="sync" />
  <invoke name="createPartyContactMech" mode="sync" />
</group>
```

```
<request-map uri="createCreditCardAndPostalAddress">
  <security https="true" auth="true" />
  <event type="service" path=""
      invoke="createCreditCardAndAddress" />
  <response name="success" type="request" value="finalizeOrder" />
  <response name="error" type="view" value="billsetting" />
</request-map>
```

4.5.1 Group element

Attribute Name	Required?	Description
name	Yes	The name of the service this action will invoke.
send-mode	No	The mode in which the service(s) should be invoked. The options are: none, all, first-available, random, or round-robin. The default is all.

Table 36 – Group element attributes

4.5.2 Invoke element

Attribute Name	Required?	Description
name	No	The name of the service this action will invoke.
mode	Yes	The mode in which this service should be invoked. Can be sync or async. Note that async actions will not update the context even when set to true.
result-to-contex	No	Should the results of the action service update the main service's context. Default false.

Table 37 – Invoke element attributes

5 Internationalization

As would be expected OFBiz supports multiple languages for its user-interfaces. The User interface texts are defined in XML files usually stored in the config folder within each component, as shown below:

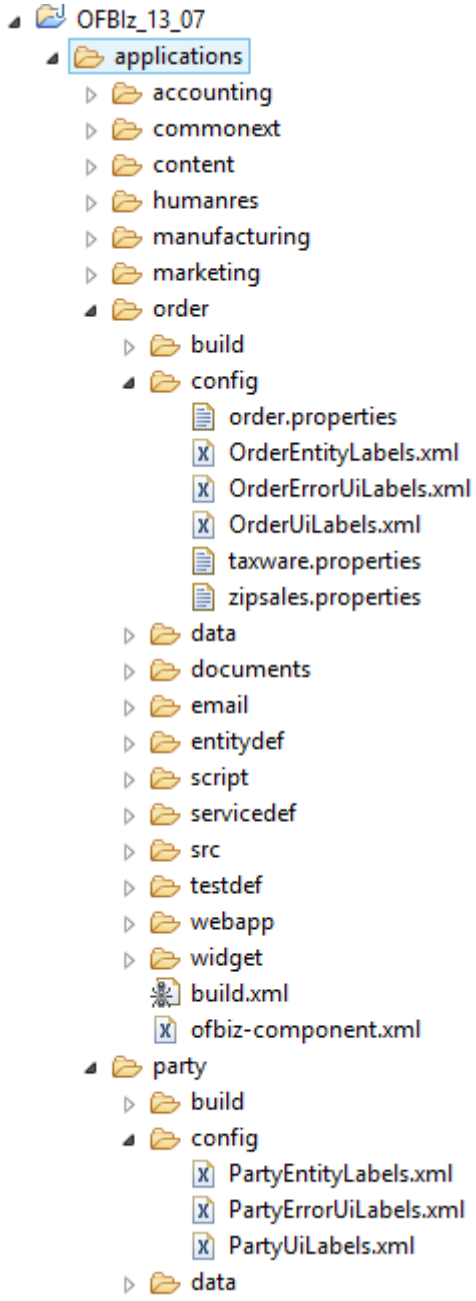


Figure 28 – OFBiz internationalization file structure

The common application texts that can be shared between components are defined in the framework/common/config folder. The texts are identified by a key that has a value for each of the supported languages.

The following XML excerpt shows the definition of the possible values for the “CommonSave” property, which defines the common value for the “Save” text:

```
<property key="CommonSave">
   <value xml:lang="ar">بيان</value>
   <value xml:lang="cs">Uložit</value>
   <value xml:lang="de">Speichern</value>
   <value xml:lang="en">Save</value>
   <value xml:lang="es">Guardar</value>
   <value xml:lang="fr">Enregistrer</value>
   <value xml:lang="hi-IN">□□□□□□□□ □□□</value>
   <value xml:lang="it">Salva</value>
   <value xml:lang="ja">保存</value>
   <value xml:lang="nl">Opslaan</value>
   <value xml:lang="pt">Guardar</value>
   <value xml:lang="pt-BR">Salvar</value>
   <value xml:lang="ro">Salveaza</value>
   <value xml:lang="ru">Сохранить</value>
   <value xml:lang="th">บันทึก</value>
   <value xml:lang="vi">Lưu</value>
   <value xml:lang="zh">保存</value>
   <value xml:lang="zh-CN">保存</value>
   <value xml:lang="zh-TW">儲存</value>
</property>
```

The following rules are applied when choosing the correct value to display:

- When the requested language is a country specification (e.g. Portuguese of Portugal) and a value is available for that specification then the value of the specification will be displayed;
- When the requested language is a country specification and only the non-specific language is available then the value for the non-specific language will be displayed;
- When the requested language is not defined but English is defined, the English value will be displayed;
- When no language is defined for the property the name of the property will be displayed;

5.1 Using texts in Mini-Lang

The property-map element provides access to OFBiz text resources by mapping all properties of a given resource to a specified map variable. This variable can then be used within a set element to retrieve the value of an existing property to another variable, e.g.:

```
<screen name="main-decorator">
  <section>
    <actions>
      <property-map resource="PartyUiLabels" map-name="uiLabelMap"
        global="true" />
      <set field="layoutSettings.companyName"
        from-field="uiLabelMap.PartyCompanyName"
        global="true" />
```

5.2 Using texts in Groovy

Accessing OFBiz text resources in a Groovy script is as simple as invoking the getResourceBundleMap from the available UtilProperties java class. The following code creates a map with all properties from the “PartyUiLabels” resource and then sets the variable “label” with the value of the PartyProfile resource property.

```
uiLabelMap = UtilProperties.getResourceBundleMap("PartyUiLabels",
      locale);
label = uiLabelMap.PartyProfile;
```

Alternatively, if only one property value is needed from the resource the getMessage method can be used instead.

```
UtilProperties.getMessage(resource,
      "PartyCannotLinkPartyToItSelf", locale)
```

5.3 Using texts in Java

Accessing text resources in Java events or services is done in the same way as it was described in the Groovy section since Groovy scripts makes use of Java classes.

6 Mini-Language

Mini-Lang is an XML defined script that is parsed and executed by the OFBiz framework. Its goal is to make repetitive tasks, like CRUD operations, simpler to develop. Since the logic is defined in XML it is somewhat easier to understand. Mini-Lang also has other advantages when compared to defining logic in Java besides being easier to write and understand. Methods defined using Mini- Lang do not need to be compiled, meaning that we can change the logic of our code when the website is running and a simple page-refresh is enough for our new code to be executed. One of the disadvantages of Mini-Lang is the debug capability, since it is defined using XML it does not support the line-by-line debug that is commonly used in Java. Taking into account this advantages and disadvantages, Mini-Lang is commonly used for simple services and events because even though Mini-Lang could be used to code complex services, the time that could take to debug the service, as well as understanding it, would exceed the time to develop it in Java.

Mini-Lang is used in OFBiz for three different purposes:

- Defining service logic;
- Defining event logic;
- Preparing data for screen widgets;

Part of the simplicity of using Mini-Lang comes from the fact that there is no need to declare variables, for example, if we need to set the value of a given variable we would simply write the following XML:

```
<set field="currentOrganizationPartyId" from-
      field="parameters.organizationPartyId" />
```

The above XML sets the field “currentOrganizationPartyId” with the value from the field “organizationPartyId” of the context parameters object. Alternatively, it is possible to set the variable with a specific value:

```
<set field="containsEmptyFields" type="Boolean" value="true" />
```

If the variable was already declared, then its value will be changed, if not, it will be declared and given the specified value.

6.1 Context Variables

In Mini-Lang we also have access to a set of “context” variables, e.g. service parameters. The following list enumerates the available context parameters. Some of the following parameters are only available on screen widgets and events.

- globalContext
    - is an instance of org.ofbiz.base.util.collections.MapStack
    - This is a context variable that can be used within all nested screens
- nullField
    - is an instance of org.ofbiz.entity.GenericEntity$NullField
    - This object is used to check for null values in database fields.
- availableLocales
    - is an instance of java.util.List

    - Contains the list of available locales supported
- locale
    - is an instance of java.util.Locale
    - Contains the current locale. This value is English by default and can be changed at any time. The current locale is the one that will be used to retrieve application texts. If a given text is not defined for the specified locale then the English value will be presented.
- delegator
    - is an instance of org.ofbiz.entity.GenericDelegator
    - Used for communicating with the database.
- dispatcher
    - is an instance of org.ofbiz.service.GenericDispatcher
    - Used to call OFBiz services.
- security
    - is an instance of org.ofbiz.security.OFBizSecurity
    - Helper object that provides a number of methods useful to execute permission checks and other security related tasks.
- userLogin
    - is an instance of org.ofbiz.entity.GenericValue
    - Used to retrieve information about the logged in user.
    - This object is not always available since the user is not required to be logged in to visit a page, e.g. the e-commerce module.
- parameters
    - is an instance of java.util.Map
    - Is a key-value pair collection that contains the service/request parameters, request attributes, session attributes and ContextServlet attributes.

6.2 Creating a Simple Service

Services defined using Mini-Lang should be placed in a <Module>Services.xml file within the scripts folder of the module.

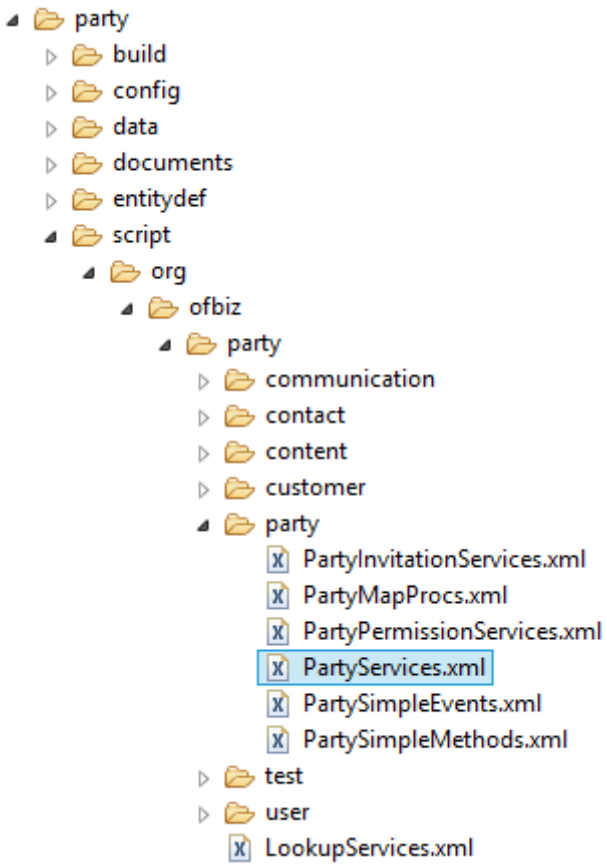


Figure 29 – OFBiz service definition file location

The following XML defines a simple service that is used to create a new Party Role.

```
<simple-method method-name="createPartyRole"
  short-description="Create Party Role">
  <entity-one entity-name="PartyRole" value-field="partyRole" />
  <if-empty field="partyRole">
    <make-value entity-name="PartyRole" value-field="newEntity" />
    <set-pk-fields map="parameters" value-field="newEntity" />
    <create-value value-field="newEntity" />
  </if-empty>
</simple-method>
```

The “entity-one” element will select a PartyRole entity with the primary key specified in the partyRole field. If no entity is found a new one will be created. The “make-value” element creates a new PartyRole entity in the newEntity variable. The “set-pk-fields” will map the fields from the “parameters” context variable to the newEntity variable and then the newly created entity will be stored in the database using the “create-value” element.

The simple method needs to be registered as a service in the services.xml file, as follows:

```
<service name="createPartyRole" engine="simple"
  location="component://party/script/org/ofbiz/
    party/party/PartyServices.xml"
  invoke="createPartyRole" auth="true">
  <description>Create a Party Role (add a Role to a Party).
    The logged in user must have PARTYMGR_CREATE or have
    permission to change the role of this partyId</description>
  <permission-service service-name="partyRolePermissionCheck"
    main-action="CREATE" />
  <attribute name="partyId" type="String" mode="IN"
    optional="true" />
  <attribute name="roleTypeId" type="String" mode="IN"
    optional="false" />
</service>
```

The service can then be invoked from the controller, as follows:

```
<request-map uri="addrole">
  <security https="true" auth="true" />
  <event type="service" path="" invoke="createPartyRole" />
  <response name="success" type="view" value="viewprofile" />
  <response name="error" type="view" value="viewprofile" />
</request-map>
```

6.3 Creating a Simple Event

Simple methods used as simple events should be placed a <Module>Events.xml file within the scripts folder of the module.

The following simple method will only set the value for the parameters of the simple method “createUser” that will then be invoked using the “call-simple-method” element.

```
<simple-method method-name="createCustomer"
  short-description="Create Customer" login-required="false">
  <set field="require_email" value="true" />
  <set field="require_phone" value="false" />
  <set field="require_login" value="true" />
  <set field="create_allow_password" value="true" />
  <set field="parameters.roleTypeId" value="CUSTOMER" />
  <call-simple-method method-name="createUser" />
</simple-method>
```

The simple method can be invoked from the controller as follows:

```
<request-map uri="createCustomer">
  <security https="true" auth="true" />
  <event type="simple"
    path="component://party/script/org/ofbiz/
      party/user/UserEvents.xml"
    invoke="createCustomer" />
  <response name="success" type="view" value="viewprofile" />
  <response name="error" type="view" value="NewCustomer" />
</request-map>
```

6.4 Variable Operations

6.4.1 Property-map

The property-map element is used to retrieve all properties (along with its values) from an OFBiz resource.

Name	Type	Requirements	Description
Resource	Constant	Required	The name of the text resource.
Map-name	Expression	Re	The name of the variable to store the values.

Table 38 – Property-map elemetn attributes

6.4.1.1 Usage

```
<property-map resource="CommonUiLabels" map-name="uiLabelMap"
      global="true"/>
```

6.4.2 Property-to-field

The property-to-field element is used to retrieve the value of a single property from an OFBiz resource.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field to set, the target of the assignment.	
Resource	constant, ${expression}	Required	The name of a properties resource. Can be a file on the classpath or a resource defined in the SystemProperty entity.	
Property	constant, ${expression}	Required	The property key.	
Default	constant, ${expression}	Optional	The default value to use if the property value is null or empty.	
no-locale	Constant	Optional	Suppress property value localization. The user's/system locale will be ignored when retrieving the property value.	Default is "false".

arg-list	Expression	Optional	The name of an argument list to be used with a formatting string. The argument list is applied to the property value.	The attribute does nothing if the argument list is not found. See the java.text.MessageFormat class for more information.

Table 39 – Property-to-field element attributes

6.4.2.1 Usage

```
<property-to-field resource="PartyUiLabels"
      property="PartyPermissionErrorPartyId" field="failMessage"/>
```

6.4.3 Set

The set element is used to assign a value to given field. The value can be an expression, script or a constant value.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field to set, the target of the assignment.	
From	expression, script	required if *value* is empty	An expression or script that returns an object or null.	The script must be prefixed with the script language followed by a colon (":").
Value	constant+expr	required if *from* is empty	A constant value.	Default type = "java.lang.String".
Default	constant, ${expression}	Optional	A default value that is used when the from attribute evaluates to null or empty.	
Type	Constant	Optional	The Java data type of *field*.	"NewList" will create a new java.util.List, "NewMap" will create a new java.util.Map. Otherwise, the attribute must contain a valid Java class name.
Locale	constant, ${expression}	Optional	If a locale dependent type is passed allows	If a locale dependent type is passed allows to i18n the value.

			to i18n the value.	
set-if-null	Constant	Optional	Controls if *field* can be set to null.	Defaults to "false".
set-if-empty	Constant	Optional	Controls if *field* can be set to an empty value. The meaning of "empty" depends on the Java data type.	D efaults to "true".

Table 40 – Set element attributes

6.4.3.1 Usage

```
<set field="inputFields" from-field="parameters"/>
<set field="orderBy" value="partyId"/>
<set field="entityName" value="PartyNameView"/>
```

6.5 Database Access

6.5.1 Entity-one element

The entity-one element is used to find an entity by its primary-key. This element returns a single GenericValue instance, if the entity is found or null if nothing is found.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	required	The name of the entity to search in.	A runtime exception will be thrown if the entity does not exist.
value-field	expression	required	The name of the field that will contain the entity value.	
use-cache	constant, ${expression}	optional	Use the entity cache.	Defaults to "false".
auto-field-map	constant, ${expression}	optional	Look for all primary key field names in the current context as well as in the parameters map.	Defaults to "true".

Table 41 – Entity-one element attributes

6.5.1.1 Usage

```
<entity-one entity-name="Person" value-field="person"/>
```

6.5.2 Entity-and element

The entity-and element is used to find an entity by specifying a set of filters that will be concatenated by an “and” operator. This element expects a field-map sub-element used to specify the filter criteria and returns a list of GenericValue instances if any entity is found.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	required	The name of the entity to search in.	A runtime exception will be thrown if the entity does not exist.
list	expression	required	The name of the field that will contain the result list.	
filter-by-date	constant, ${expression}	optional	Filter the result list by the current date.	The entity must have "fromDate" and "thruDate" fields. Defaults to "false".
distinct	constant, ${expression}	optional	Filter the results so that all of them are unique.	Defaults to "false".
use-cache	constant, ${expression}	optional	Use the entity cache.	Defaults to "false".
delegator-name	constant, ${expression}	optional	Overrides the current delegator by specifying a delegator name.	

Table 42 – Entity-and element attributes

6.5.2.1 Usage

```
<entity-and entity-name="PartyNameHistory" list="partyNameHistoryList">
      <field-map field-name="partyId" from-field="parameters.partyId"/>
      <order-by field-name="-changeDate"/>
</entity-and>
```

6.5.3 Entity-condition element

The entity condition element is used to find entities by a specific criteria. This element is similar to the entity-and element but provides greater flexibility regarding the filter criteria.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	required	The name of the entity to search in.	A runtime exception will be thrown if the entity does not exist.
list	expression	required	The name of the field that will contain the result list.	
filter-by-date	constant, ${expression}	optional	Filter the result list by the current date.	The entity must have "fromDate" and "thruDate" fields. Defaults to "false".
distinct	constant, ${expression}	optional	Filter the results so that all of them are unique.	Defaults to "false".
use-cache	constant, ${expression}	optional	Use the entity cache.	Defaults to "false".
delegator-name	constant, ${expression}	optional	Overrides the current delegator by specifying a delegator name.	

Table 43 – Entity-condition element attributes

The condition-list specifies a list of conditions that will compose the criteria and will be combined using either the “and” or the “or” operator. By default the “and” operator is used.

6.5.3.1 Usage

```
<entity-condition list="prodCatalogCategoryList" entity-
  name="ProdCatalogCategory" filter-by-date="true">
  <condition-list combine="and">
    <condition-expr field-name="productCategoryId"
      from-field="parameters.productCategoryId"/>
      <condition-list combine="or">
        <condition-expr field-name="prodCatalogCategoryTypeId"
            value="PCCT_VIEW_ALLW"/>
        <condition-expr field-name="prodCatalogCategoryTypeId"
            value="PCCT_PURCH_ALLW"/>
    </condition-list>
  </condition-list>
</entity-condition>
```

6.5.4 Entity-count element

The entity-count element is used to retrieve a count of entities that match a given set of conditions. It supports a condition list similar to the entity-condition element.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	required	The name of the entity to count in.	A runtime exception will be thrown if the entity does not exist.
count-field	expression	required	Name of the field that will contain the count result.	
delegator-name	constant, ${expression}	optional	Overrides the current delegator by specifying a delegator name.	

Table 44 – Entity-count element attributes

6.5.4.1 Usage

```
<entity-count entity-name="ReturnAdjustment" count-field="returnCount">
  <condition-expr field-name="orderAdjustmentId" operator="equals"
      from-field="orderAdjustment.orderAdjustmentId"/>
</entity-count>
```

6.5.5 Get-related-one element

The get-related-one element is used to retrieve an entity that is related to another entity base on the declared relations of the primary entity (e.g. an entity that is referenced using a foreign key).

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
relation-name	constant, ${expression}	Required	The name of the entity value relation to use.	Script execution will halt if the relation does not exist.
to-value-field	expression	required	The name of the field that will contain the related entity value.	
use-cache	constant, ${expression}	Optional	Use the entity cache.	Defaults to "false".

Table 45 – Get-related-one element attributes

6.5.5.1 Usage

```
<entity-one entity-name="CustRequest" value-field="custRequest"/>
<get-related-one value-field="custRequest"
      relation-name="StatusItem" to-value-field="statusItem"/>
```

6.5.6 Get-related

The get-related element is similar to the get-related-one as it is also based on the declared relations of an entity but instead of retrieving only one value it is used to retrieve a list of related entities (e.g. entities that have N records related to 1 entity).

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
relation-name	constant, ${expression}	Required	The name of the entity value relation to use.	Script execution will halt if the relation does not exist.
List	Expression	Required	The name of the field that will contain the result list.	
Map	Expression	required	The name of the field containing a map that will be used for the search.	The map name/value pairs will be combined using a boolean AND. All entity values will be found if this attribute is empty.
order-by-list	Expression	Optional	The name of the field containing a list that contains field names that you want the operation to order the results by.	Each entry in the list is a field name. The field name can be preceded by a plus or a minus sign to specify an ascending or descending sort for that field. The default is ascending sort.
use-cache	constant, ${expression}	Optional	Use the entity cache.	Defaults to "false".

Table 46 – Get-related element attributes

6.5.6.1 Usage

```
<get-related value-field="partyContactMech"
      relation-name="PartyContactMechPurpose"
      list="partyContactMechPurposes"/>
```

6.5.7 Make-value

The make-value element is used to create a new entity that exists only in memory.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	Required	The name of the entity.	A runtime exception will be thrown if the entity does not exist.
value-field	Expression	required	The name of the field that will contain the new entity value.	
Map	Expression	Required	The name of the field that contains a map that is used to initialize the entity value fields.	Map values whose key matches one of the entity value fields will be copied to the corresponding entity value field.

Table 47 – Make-value element attributes

6.5.7.1 Usage

```
<make-value entity-name="PartyRole" value-field="newEntity"/>
<set-pk-fields map="parameters" value-field="newEntity"/>
<create-value value-field="newEntity"/>
```

6.5.8 Clone-value

The clone-value element is used to create a copy of an existing entity value.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field that contains the entity value to be copied.	The operation does nothing if the entity value is not found.
new-value-field	Expression	Required	The name of the field that will contain the new entity value.	

Table 48 – Clone-value element attributes

6.5.8.1 Usage

```
<find-by-primary-key entity-name="Invoice" map="lookupPKMap"
      value-field="lookedUpValue"/>
<clone-value value-field="lookedUpValue"
      new-value-field="savedValue"/>
```

6.5.9 Create-value

The create-value element is used to create a new record on the OFBiz database.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field that contains the entity value to be created.	The operation generates an error and halts script execution if the entity value is not found.
do-cache-clear	Constant	Optional	Clear the cache.	Defaults to "true".
or-store	Constant	Optional	Store the entity value if it already exists.	Defaults to "false". If set to "false", an error will result if an instance of the entity value exists in the data source with the same primary key.

Table 49 – Create-value element attributes

6.5.9.1 Usage

```
<make-value entity-name="PartyRole" value-field="newEntity"/>
<set-pk-fields map="parameters" value-field="newEntity"/>
<create-value value-field="newEntity"/>
```

6.5.10 Store-value

The store-value element is used to update an existing record on the OFBiz database.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist. Script execution will halt if the entity value does not exist in the data source.
do-cache-clear	constant, ${expression}	Optional	Clear the cache.	Defaults to "true".

Table 50 – Store-value element attributes

6.5.10.1 Usage

```
<entity-one entity-name="PartyClassification"
      value-field="lookedUpValue"/>
<set-nonpk-fields value-field="lookedUpValue" map="parameters"/>
<store-value value-field="lookedUpValue"/>
```

6.5.11 Remove-value

The remove-value is used to remove a record from the database.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
do-cache-clear	constant, ${expression}	optional	Clear the cache.	Defaults to "true".

Table 51 – Remove-value element attributes

6.5.11.1 Usage

```
<entity-one entity-name="PartyRole" value-field="partyRole"/>
<remove-value value-field="partyRole"/>
```

6.5.12 Remove-related

The remove-related element is used to remove all entities related with the provided entity. For one-to-one relationships only one record will be removed, for one-to-many relationships all records will be removed.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
relation-name	constant, ${expression}	Required	The name of the entity value relation to use.	Script execution will halt if the relation does not exist.
do-cache-clear	constant, ${expression}	Optional	Clear the cache.	Defaults to "true".

Table 52 – Remove-related element attributes

6.5.12.1 Usage

```
<entity-one entity-name="Requirement" value-field="requirement"
      auto-field-map="true"/>
<check-errors/>
<remove-related value-field="requirement"
      relation-name="RequirementRole"/>
<remove-related value-field="requirement"
      relation-name="RequirementCustRequest"/>
<remove-value value-field="requirement"/>
```

6.5.13 Remove-by-and

The remove-by-and element uses a map of name/value pairs that will be used to generate filters to constrain the removal. These filters will be combined using the AND operator.

Name	Type	Requirements	Description	Notes
entity-name	constant, ${expression}	Required	The name of the entity to search in.	A runtime exception will be thrown if the entity does not exist.
Map	expression	Required	The name of the field containing a map that will be used for the constraint.	The map name/value pairs will be combined using a boolean AND. If the map is missing or empty, then all entity values will be removed.
do-cache-clear	constant, ${expression	Optional	Clear the cache.	Defaults to "true".

Table 53 – Remove-by-and element attributes

6.5.13.1 Usage

```
<set field="paymentApplicationMap.invoiceId"
      from-field="parameters.invoiceId"/>
<set field="paymentApplicationMap.invoiceItemSeqId"
      from-field="parameters.invoiceItemSeqId"/>
<remove-by-and entity-name="PaymentApplication"
      map="paymentApplicationMap"/>
```

6.5.14 Set-pk-fields

The set-pk-fields element is used to copy the primary key values from a map to an entity value.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
Map	Expression	Required	The name of the field that contains a map that is used to initialize the entity value pk fields.	Map values whose key matches one of the entity value fields will be copied to the entity value. A runtime exception will be thrown if the

				map does not exist.
set-if-null	constant, ${expression}	Set entity value fields that are null or empty.	Defaults to "true".	

Table 54 – Set-pk-fields element attributes

6.5.14.1 Usage

```
<make-value entity-name="PartyRole" value-field="newEntity"/>
<set-pk-fields map="parameters" value-field="newEntity"/>
```

6.5.15 Set-non-pk-fields

The set-non-pk-fields element is used to copy the non-primary key values from a map to an entity value.

Name	Type	Requirements	Description	Notes
value-field	Expression	Required	The name of the field containing the entity value.	A runtime exception will be thrown if the entity value does not exist.
Map	Expression	Required	The name of the field that contains a map that is used to initialize the entity value non-pk fields.	Map values whose key matches one of the entity value fields will be copied to the entity value. A runtime exception will be thrown if the map does not exist.
set-if-null	constant, ${expression}	Optional	Set entity value fields that are null or empty.	Defaults to "true".

Table 55 – Set-non-pk-fields element attributes

6.5.15.1 Usage

```
<make-value value-field="newEntity"
      entity-name="PartyAcctgPreference"/>
<set-pk-fields map="parameters" value-field="newEntity"/>
<set-nonpk-fields map="parameters" value-field="newEntity"/>
<create-value value-field="newEntity"/>
```

6.5.16 Store-list

The store-list element is used to update the value of all records on the list that exist on the database. Values that are present on the list and do not exist on the database will be created.

Name	Type	Requirements	Description	Notes
List	Expression	Required	The name of the field that contains the list of entity values.	A runtime exception will be thrown if the list does not exist.
do-cache-clear	constant, ${expression}	Optional	Clear the cache.	Defaults to "true".

Table 56 – Store-list element attributes

6.5.16.1 Usage

```
<iterate entry="categoryEntity" list="categoryEntities">
    <clone-value value-field="categoryEntity"
      new-value-field="newCategoryEntity"/>
    <set field="newCategoryEntity.productCategoryId"
      from-field="productCategoryIdTo"/>
    <field-to-list field="newCategoryEntity"
      list="entitiesToStore"/>
</iterate>
<store-list list="entitiesToStore"/>
```

6.6 Logical Operators

6.6.1 If-validate-method

The if-validate-method is used to invoke a static method and if that method returns true then the logic contained within the element will be executed. If the static method returns false the logic within the else sub-element will be executed, if one exists.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field to use as the method argument.	The field object will be converted to a string before the method call. An empty string will be used if the field object is not found.
Method	Constant	Required	The name of the static method that will be called to validate the field.	Must be a static method that takes a single java.lang.String parameter and return a boolean.
Class	Constant	Optional	The name of the Java class that contains the validation method.	Defaults to "org.ofbiz.base.util.UtilValidate".

Table 57 – If-validate-method element attributes

6.6.1.1 Usage

```
<if-validate-method field="parameters.emailAddress"
      method="isEmail">
  <else>
    <add-error>
      <fail-property resource="PartyUiLabels"
        property="PartyEmailAddressNotFormattedCorrectly"/>
    </add-error>
  </else>
</if-validate-method>
```

6.6.2 If-compare

The if-compare element is used to execute a block of code if the condition specified in its attributes evaluates to true, otherwise the logic contained within the else sub-element will be executed, if one exists.

Name	Type	Requirements	Description	Notes
Field	expression	Required	The name of the field that will be compared. The l-value.	
Operator	Constant	Required	The comparison operator. The operator describes the l-value compared to the r-value.	Valid values are: "equals", "not-equals", "less", "less-equals", "greater", "greater-equals", "contains", "is-null", "is-not-null", "is-empty".
Value	constant+expr	ignored when operator attribute equals "is-null", "is-not-null", or "is-empty" - required otherwise	The value that the field will be compared to. The r-value.	
Type	Constant	Optional	The Java data type. Both values will be converted to this type before comparison.	Attribute must contain a valid Java class name. Invalid when the operator attribute value equals "contains".
Format	constant, ${expression}	Optional	Format used for type conversions.	Valid only when the type attribute is not empty.

Table 58 – If-compare element attributes

6.6.2.1 Usage

```
<if-compare field="parameters.lastNameFirst"
      operator="equals" value="Y">
  <set field="fullName"
      value="${person.personalTitle} ${person.lastName},
       ${person.firstName} ${person.middleName} ${person.suffix}"/>
  <else>
    <set field="fullName"
      value="${person.personalTitle} ${person.firstName}
      ${person.middleName} ${person.lastName} ${person.suffix}"/>
  </else>
</if-compare>
```

6.6.3 If-compare-field

The if-compare-field is used to compare the value of two fields using the provided operator. If the condition evaluates to true then the code block within the element will be executed, otherwise the logic within the else sub-element will be executed, if one exists.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field that will be compared. The l-value.	
Operator	Constant	Required	The comparison operator. The operator describes the l-value compared to the r-value.	Valid values are: "equals", "not-equals", "less", "less-equals", "greater", "greater-equals", "contains".
to-field	Expression	Required	The name of the field that *field* will be compared to. The r-value.	
Type	Constant	Optional	The Java data type. Both values will be converted to this type before comparison.	Attribute must contain a valid Java class name. Invalid when the operator attribute value equals "contains".
Format	constant, ${expression}	Optional	Format used for type conversions.	Valid only when the type attribute is not empty.

Table 59 – If-compare-field element attributes

6.6.3.1 Usage

```
<if-compare-field to-field="userLogin.partyId"
      field="parameters.partyId" operator="equals">
    <set field="hasPermission" type="Boolean" value="true"/>
    <field-to-result field="hasPermission"/>
    <else>
        <set field="altPermission" value="PARTYMGR_PCM"/>
        <call-simple-method method-name="basePermissionCheck"/>
    </else>
</if-compare-field>
```

6.6.4 If-empty

The if-empty element is used to test if the value of a given field is null or empty. If the condition evaluates to true, otherwise the logic contained within the else sub-element will be executed, if one exists.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field to test.	Object to test must be a collection, string, or a class that implements org.ofbiz.base.lang.IsEmpty.

Table 60 – If-empty element attributes

6.6.4.1 Usage

```
<if-empty field="parameters.asOfTimestamp">
     <now-timestamp field="nowTimestamp"/>
 <else>
     <set from-field="parameters.asOfTimestamp"
      field="nowTimestamp"/>
 </else>
</if-empty>
```

6.6.5 If-not-empty

The if-not-empty element is used to test if the value of a given field is not null nor empty. If the condition evaluates to true, otherwise the logic contained within the else sub-element will be executed, if one exists.

Name	Type	Requirements	Description	Notes
Field	Expression	Required	The name of the field to test.	Object to test must be a collection, string, or a class that implements org.ofbiz.base.lang.IsEmpty.

Table 61 – If-not-empty element attributes

6.6.5.1 Usage

```
<if-not-empty field="entityValue">
  <set field="aggregatedPartyAcctgPref.${entityKey}" from-
      field="entityValue"/>
  <else>
    <set field="containsEmptyFields" type="Boolean" value="true"/>
  </else>
</if-not-empty>
```

6.7 Security Validations

6.7.1 Check-permission

The check-permission element validates if the user haves the specified permission and adds an error message to the error messages list if the user does not have the required permission.

Name	Type	Requirements	Description	Notes
Permission	constant, ${expression}	Required	The permission to check.	
Action	constant, ${expression}	Optional	The action to be performed (permission scope).	Examples: "_ADMIN", "_CREATE", "_UPDATE".
error-list-name	Constant	Optional	The name of a list that will contain the message.	Defaults to "error_list".

Table 62 – Check-permission element attributes

6.7.1.1 Usage

```
<check-permission permission="PARTYMGR" action="_UPDATE">
    <fail-property resource="PartyUiLabels"
      property="PartyPermissionErrorForThisParty"/>
</check-permission>
```

6.8 Calling Operations

6.8.1 Call-class-method

The call-class-method element is used to invoke a static method on a java class. This element contains two sub-elements: “string” and “field” used to specify the arguments to be sent to the method. The field sub-element requires the value to be sent and the type while the string sub-element assumes that the type of the value to be sent is a string.

Name	Type	Requirements	Description
class-name	Constant	required	The name of the class containing the static method.
method-name	Constant	Required	The name of the static method to call.
ret-field	Expression	Optional	The name of the field to put the method return value in.

Table 63 – Call-class-method element attributes

6.8.1.1 Usage

```
<call-class-method class-name="org.ofbiz.base.util.UtilDateTime"
      method-name="getYearStart"
      ret-field="curYearFiscalStartDate">
    <field field="nowTimestamp" type="java.sql.Timestamp"/>
    <field field="partyAcctgPreference.fiscalYearStartDay"
      type="java.lang.Number"/>
    <field field="partyAcctgPreference.fiscalYearStartMonth"
      type="java.lang.Number"/>
    <field field="zeroLong" type="java.lang.Number"/>
</call-class-method>
```

6.8.2 Call-service

The call-service element is used to invoke an OFBiz service.

Name	Type	Requirements	Description	Notes
service-name	constant, ${expression}	Required	The name of the service to invoke.	
in-map-name	Expression	Optional	The name of a field containing a map that will be used for the service's IN parameters.	
include-user-login	Constant	Optional	Include the current UserLogin entity value in the called service IN parameters.	Defaults to "true".
break-on-error	Constant	Optional	Halt script execution if the called service returns an error.	Defaults to "true".
error-code	Constant	Optional	The error code returned by the called service.	Defaults to the enclosing <simple-method> "default-error-code" attribute value.
success-code	Constant	Optional	The success code returned by the called service.	Defaults to the enclosing <simple-method> "default-success-code" attribute value.
require-new-transaction	Constant	Optional	Require a new transaction for the called service.	Defaults to "false".
transaction-timeout	Constant	Optional	The timeout for the new transaction, in seconds.	Defaults to the value set in the called service's definition.

Table 64 – Call-service element attributes

The call-service has the following set of sub-elements that are used to retrieve the service output parameters:

- <results-to-map>
- <result-to-field>
- <result-to-request>
- <result-to-session>
- <result-to-result>

6.8.2.1 Usage

```
<call-service service-name="createContent"
      in-map-name="createContentMap" break-on-error="false">
     <result-to-field result-name="contentId" field="contentId"/>
</call-service>
```

6.8.3 Call-simple-method

The call-simple-method element can be used to invoke a Mini-Language method.

Name	Type	Requirements	Description	Notes
method-name	Constant	Required	The name of the <simple-method> element.	
xml-resource	Constant	Optional	The location of the <simple-method> file.	Defaults to current file.
scope	Constant	Optional	The memory scope to use: "inline" or "function".	When set to "inline", existing variables can be modified by the called script. When set to "function", existing variables are protected from modification, and the called script returns values via the `<field-to-result>` element. Defaults to "inline".

Table 65 – Call-simple-method element attributes

6.8.3.1 Usage

```
<call-simple-method method-name="genericBasePermissionCheck"
      xml-resource="component://common/script/org/ofbiz/
      common/permission/CommonPermissionServices.xml"/>
```

6.8.4 Script

The script element is used to run Groovy scripts.

Name	Type	Requirements	Description	Notes
Location	Constant	required if *script* is empty	The location of the script file.	Script functions/methods can be invoked by appending a hash (#) and the function/method name.
Script	Script	required if *location* and element body are empty	A short script (scriptlet). Can be used instead of a file.	

Table 66 – Script element attributes

6.8.4.1 Usage

```
<script location="component://accounting/webapp/ap/WEB-INF/
      actions/invoices/CommissionReport.groovy"/>
```

6.9 Looping operations

6.9.1 Iterate

The iterate element executes the block of code declared within itself for each entry in the provided collection.

Name	Type	Requirements	Description	Notes
List	Expression	Required	The name of the field that contains the collection to iterate over.	The operation does nothing if the field is not found.
Entry	Expression	Required	The name of the field that will contain the current collection entry.	

Table 67 – Iterate element attributes

6.9.1.1 Usage

```
<iterate list="postalAddressBoundaries"
      entry="postalAddressBoundary">
    <get-related-one value-field="postalAddressBoundary"
      to-value-field="geo" relation-name="Geo"/>
    <field-to-list field="geo" list="geos"/>
</iterate>
```

6.9.2 Iterate-map

The iterate-map element executes the block of code declared within itself for each entry in a map.

Name	Type	Requirements	Description	Notes
Map	Expression	Required	The name of the field that contains the map to iterate over.	The operation does nothing if the field is not found.
Key	Expression	Required	The name of the field that will contain the current map entry key.	
Value	Expression	Required	The name of the field that will contain the current map entry value.	

Table 68 – Iterate-map element attributes

6.9.2.1 Usage

```
<iterate-map key="key" value="value"
      map="parameters.identifications">
    <entity-one value-field="identificationType"
      entity-name="PartyIdentificationType" use-cache="true">
        <field-map field-name="partyIdentificationTypeId"
      from-field="value"/>
    </entity-one>
    <if-not-empty field="identificationType">
        <set field="idValue"
            value="${parameters.identifications.
            ${identificationType.partyIdentificationTypeId}}"/>
        <if-not-empty field="idValue">
            <set field="partyIdentCtx.partyIdentificationTypeId"
            from-field="identificationType.
            partyIdentificationTypeId"/>
            <set field="partyIdentCtx.idValue"
                  from-field="idValue"/>
            <call-service service-name="createPartyIdentification"
                  in-map-name="partyIdentCtx"/>
        </if-not-empty>
    </if-not-empty>
</iterate-map>
```

6.9.3 Loop

The loop element is used to execute the block of code declared within itself until a maximum count is reached.

Name	Type	Requirements	Description	Notes
Count	constant, ${expression}	Required	A maximum count expression.	Must evaluate to a positive integer. Defaults to "0" (zero)
Field	Expression	Optional	The name of the field that will contain the current count value.	

Table 69 – Loop element attributes

6.9.3.1 Usage

```
<set field="loops" from-field="parameters.quantityAccepted"/>
<loop count="${loops}" field="currentLoop">
…
</loop>
```

7 Groovy

Groovy is a powerful, optionally typed and dynamic language. It integrates smoothly with any Java program, and immediately delivers to your application powerful features, including scripting capabilities, Domain-Specific Language authoring, runtime and compile-time meta-programming and functional programming. The groovy programming language can be used within OFBiz to code events or act as a replacement for the mini-language sintax within the actions element of a view. The following XML shows how to invoke a Groovy script from the actions element of a view.

```
<screen name="ReportResultDisplayInternal">
  <section>
    <actions>
      <script
        location="component://reports/webapp/reports/
            WEB-INF/actions/ReportResultDisplayInternal.groovy" />
    </actions>
    ...
```

The following Groovy code shows how we can query the database using Groovy. For this case the sintax is similar to Java because we are only accessing context objects and invoking methods of those objects, which covers the common usage scenarios.

```
import java.util.*;
import java.lang.*;
import org.ofbiz.entity.*;
import org.ofbiz.entity.condition.*;
import org.ofbiz.base.util.*;
import org.ofbiz.base.util.UtilMisc;

context.report = null;
context.reportValidationMessages = null;
context.reportQueue = null;

if (parameters.reportQueueId != null){
  context.reportQueue = delegator.findOne("ReportQueue",
      false,
      UtilMisc.toMap("reportQueueId", parameters.reportQueueId));

  if(context.reportQueue != null)
  {
    reports = context.reportQueue.getRelated("Report",
            null, null, false);

    if(reports != null && reports.size() > 0){
      context.report = reports.get(0);
    }
  }
```

```
} else {
  context.report = delegator.findOne("Report",
      false, UtilMisc.toMap("reportId", parameters.reportId));

  if(context.report != null)
  {
    reportQueues = context.report.getRelated("ReportQueue",
      null, null, false);

    if(reportQueues != null && reportQueues.size() > 0){
      context.reportQueue = reportQueues.get(0);
    }
  }
}

if(context.report != null){
  context.reportValidationMessages =
      context.report.getRelated("ReportValidationMsgs",
      null, null, false);
}
```

We chose to use Groovy in this situation because the actions element does not allow if-then-else structures and this code does not represent a service.

More details regarding the Groovy programming language and its integration with OFBiz can be found in the following URLs:

```
http://www.groovy-lang.org/
```

```
https://cwiki.apache.org/confluence/display/OFBIZ/Groovy+DSL+for+OFBiz+business+logic
```

8 Event Condition Actions (ECA)

ECA refers to an action that is executed when a given event occurs and a specific condition is met. OFBiz has two types of ECA rules: the service event condition actions (SECAs) and entity event condition actions (EECAs).

8.1 Service Event Condition Actions (SECAs)

The trigger for the SECAs are, as the name suggests, services being invoked and the condition could be checking the value of a given parameter, as conditions are optional. When the condition is met, if any, another service will be invoked.

SECAs are defined in the servicedef folder in a file named seca.xml.

```
<eca service="storeOrder" event="return">
  <condition field-name="orderTypeId"
    operator="equals" value="PURCHASE_ORDER" />
  <action service="setUnitPriceAsLastPrice" mode="sync" />
</eca>
```

The XML presented above contains the definition of an ECA rule that invokes the “setUnitPriceAsLastPrice” service before the “storeOrder” service returns a value and the “orderTypeId” equals “PURCHASE_ORDER”.

The event attribute defines when, in the service execution lifecycle, the action will be executed. The event can be executed at the following service lifecycle stages:

- auth – Before authorization
- in-validate – Before the input parameters are validated
- out-validate – Before the output parameters are validated
- invoke – Before the service is invoked
- commit – Before the transaction is committed
- return – Before the service returns
- global-commit – If the service is part of a transaction, the service will only run between the two phases of commit.
- global-rollback – If the service is part of a transaction, the service will only run after the rollback.

8.2 Entity Event Condition Action (EECAs)

The trigger for the EECAs are operations being executed on entities and the condition could be checking the value of a given field, as conditions are optional. When the condition is met, if any, a service is being invoked. EECAs are defined in the servicedef folder, in a file named eeca.xml.

```
<eca entity="OrderItem" operation="create-store" event="return">
  <condition field-name="quoteId" operator="is-not-empty" />
  <action service="checkUpdateQuoteStatus" mode="sync" />
</eca>
```

The xml presented above contains the definition of an ECA rule that invokes the “checkUpdateQuoteStatus” service when a new OrderItem is created or updated and the “quoteId” is not empty.

The operation attribute defines the operation that is being executed on the entity that will trigger the action. The operation attribute can have the following values:

- create
- store
- remove
- find
- create-store
- create-remove
- store-remove
- create-store-remove
- any

Before using any ECA on a component the eca file must be registered on the component definition file “ofbiz-component.xml” within the ofbiz module.

```
<entity-resource type="eca" reader-name="main" loader="main"
      location="entitydef/eecas.xml" />
```

The following EECA executes the service “updateFinAccountBalancesFromTrans” whenever a new FinAccountTrans entity is created and the statusId field equals “FINACT_TRNS_APPROVED”.

```
<entity-eca xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/
      dtds/entity-eca.xsd">
  <eca entity="FinAccountTrans"
      operation="create-store" event="return">
    <condition  field-name="statusId"
      operator="equals" value="FINACT_TRNS_APPROVED"/>
    <action service="updateFinAccountBalancesFromTrans"
      mode="sync"/>
  </eca>
</entity-eca>
```

8.3 ECA element

Attribute Name	Required?	Description
service	Yes	The name of the service this ECA is attached to.
event	Yes	The event on which this ECA will run can be (before): auth, in-validate, out-validate, invoke, commit, or return.
run-on-error	No	Should this ECA run if there is an error in the service (default false)

Table 70 – ECA element attributes

8.4 Condition element

Attribute Name	Required?	Description
map-name	No	The name of the service context field that contains the map that the field to be validated will come from. If not specified the field-name will be treated as a service context field name (an env-name).
field-name	Yes	The name of the map field that will be compared.
operator	Yes	Specified the comparison operator must be one of the following: less, greater, less-equals, greater-equals, equals, not-equals, or contains.
value	Yes	The value that the field will compared to. Must be a String, but can be converted to other types.
type	No	The data type to use for the comparison. Must be one of the following: String, Double, Float, Long, Integer, Date, Time, or Timestamp. If no type is specified the default will be String.
format	No	A format specifier to use when converting String objects to other data types, mainly Date, Time and Timestamp.

Table 71 – Condition element attributes

8.5 Condition-field element

Attribute Name	Required?	Description
map-name	No	The name of the service context field that contains the map that the field to be validated will come from. If not specified the field-name will be treated as a method environment field name (an env-name).
field-name	Yes	The name of the map field that will be compared.
operator	Yes	Specified the comparison operator must be one of the following: less, greater, less-equals, greater-equals, equals, not-equals, or contains.
to-map-name	No	The name of the service context field that contains the map that the field to be compared will come from. If left empty will default to the map-name, or the method environment if map-name is also unspecified.
to-field-name	No	The name of the to-map field that the main field will be compared to. If left empty will default to the field-name.
type	No	The data type to use for the comparison. Must be one of the following: String, Double, Float, Long, Integer, Date, Time, or Timestamp. If no type is specified the default will be String.
format	No	A format specifier to use when converting String objects to other data types, mainly Date, Time and Timestamp.

Table 72 – Condition-field element attributes

8.6 Action element

Attribute Name	Required?	Description
service	No	The name of the service this action will invoke.
mode	Yes	The mode in which this service should be invoked. Can be sync or async. Note that async actions will not update the context even when set to true.
result-to-context	No	Should the results of the action service update the main service's context. Default true.
result-to-result	No	Should the results of the action service update the main service's results. If true, the action service's messages will be appended to the main service's messages. Default false.
ignore-error	No	Ignore any errors caused by the action service. If true the error will cause the original service to fail. Default true.
persist	No	The action service store / run. Can be true or false. Only effective when mode is async. Default false.

Table 73 – Action element attributes

9 WebTools

The OFBiz WebTools is an OFBiz module that is intended to be used by system administrators and developers. The WebTools module contains, among others, the following tools:

This module can be accessed using the following url:

```
https://127.0.0.1:8443/webtools/control/main
```

9.1 Cache and Debug tools

Allows system administrators to view what elements are in the OFBiz caches, how much memory are they using and when they will expire. System administrators are also able to clear individual caches, as well as all of them or simply remove specific elements within a cache. It is also possible to run the garbage collection.

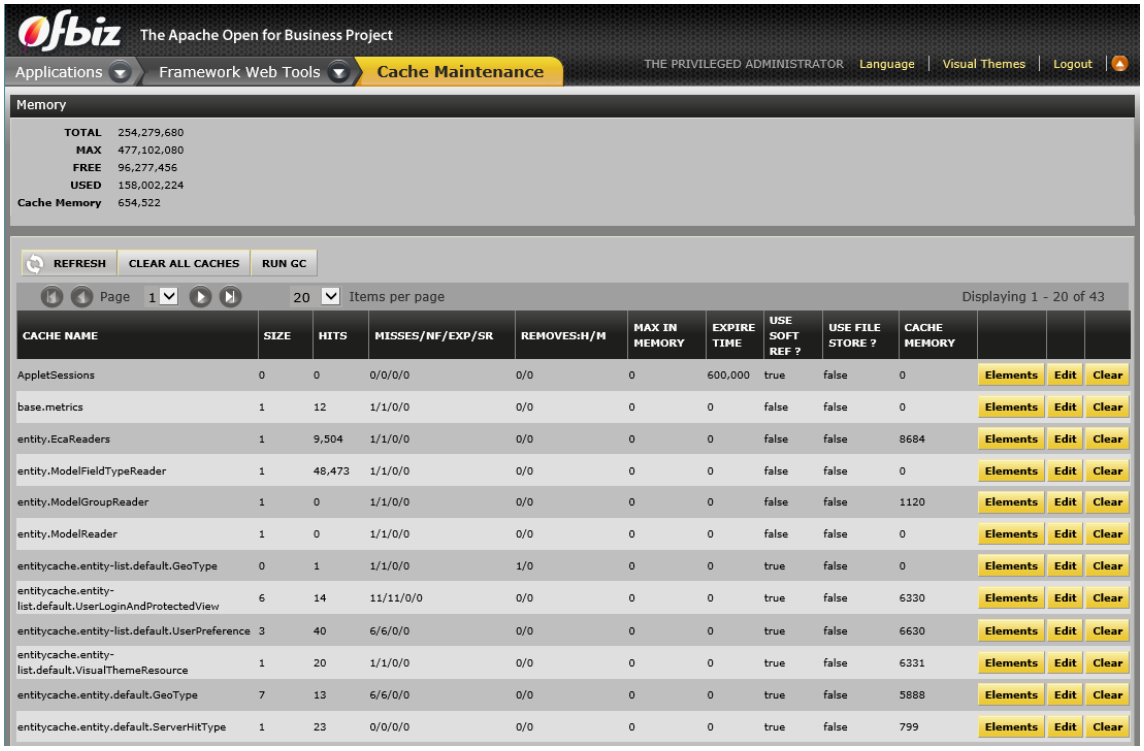


Figure 30 – OFBiz Cache Maintenance page

Another component of the Cache and Debugging tools are the Debugging management allowing system administrators to specify the level of the messages that will be logged. There is also a page that will display all logged messages.

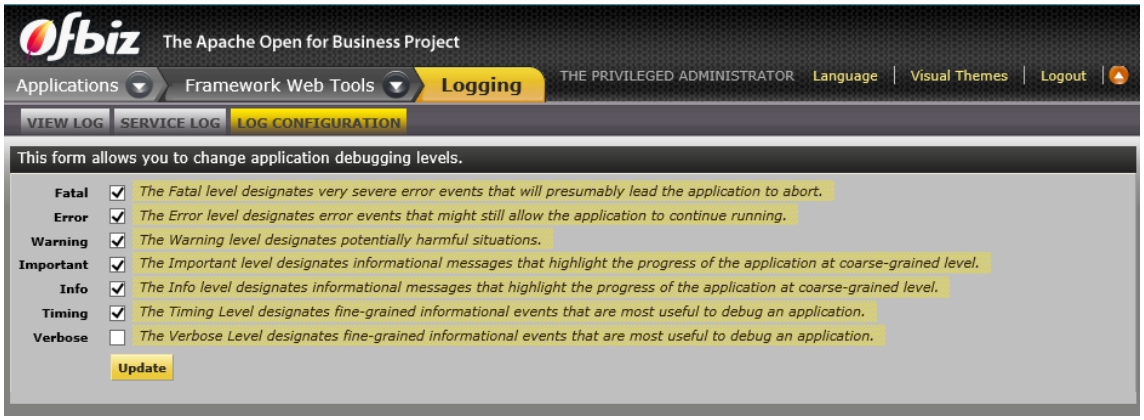


Figure 31 – OFBiz logging configuration page

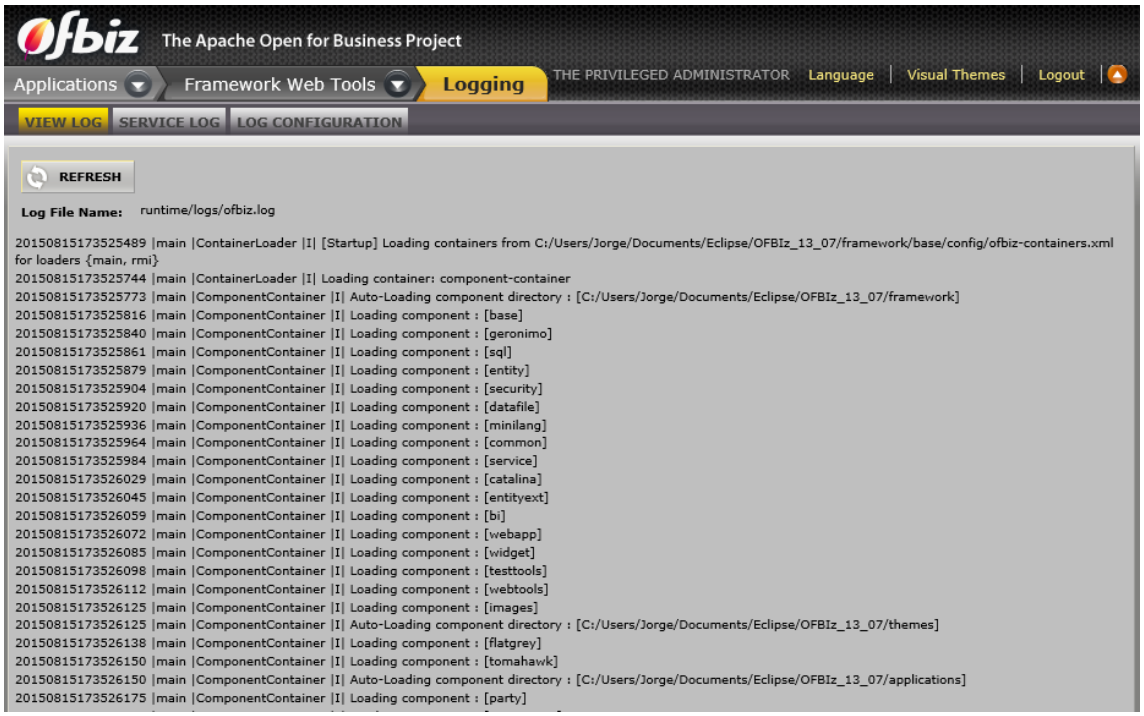
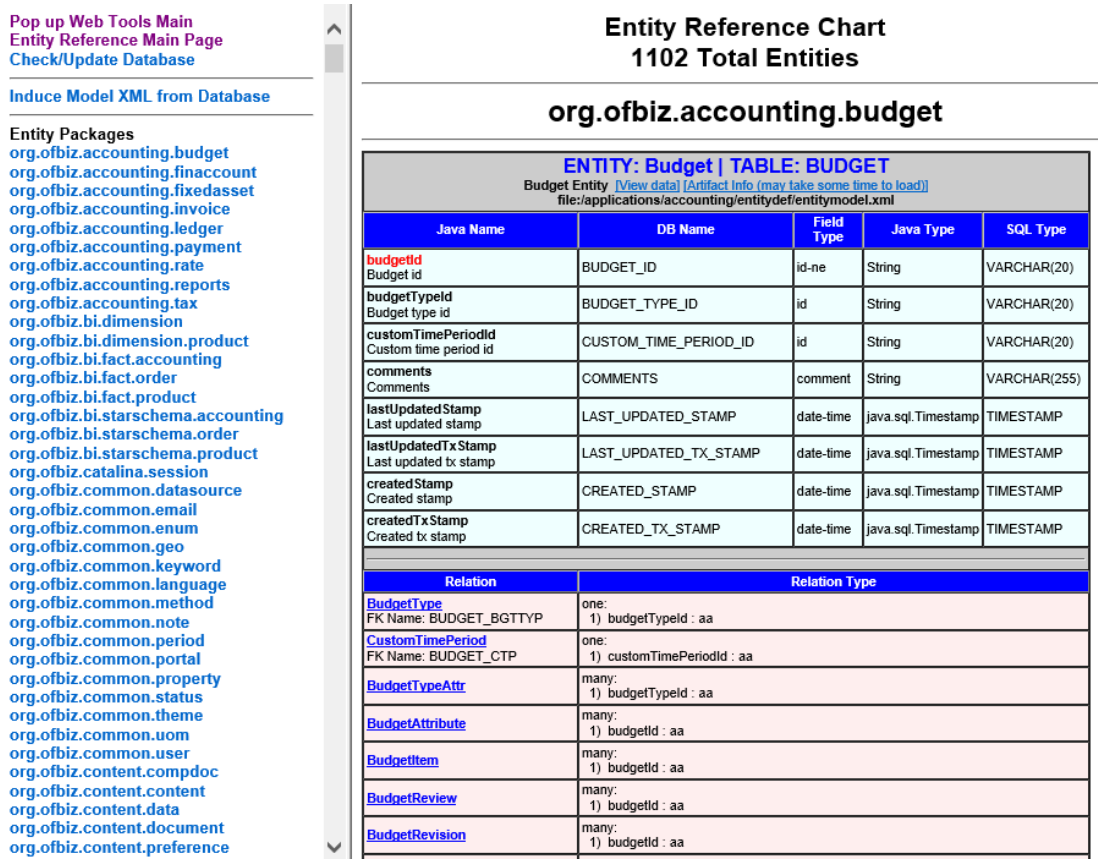


Figure 32 – OFBiz log display page

9.2 General artifact information tools

The general artifact information tools provide a reference of the OFBiz services and entities. The service reference page lists all services and allows system administrators to call a given service providing the input parameters and checking the output parameters of the service or schedule a job.



Pop up Web Tools Main
Entity Reference Main Page
Check/Update Database

Induce Model XML from Database

Entity Packages
org.ofbiz.accounting.budget
org.ofbiz.accounting.finaccount
org.ofbiz.accounting.fixedasset
org.ofbiz.accounting.invoice
org.ofbiz.accounting.ledger
org.ofbiz.accounting.payment
org.ofbiz.accounting.rate
org.ofbiz.accounting.reports
org.ofbiz.accounting.tax
org.ofbiz.bi.dimension
org.ofbiz.bi.dimension.product
org.ofbiz.bi.fact.accounting
org.ofbiz.bi.fact.order
org.ofbiz.bi.fact.product
org.ofbiz.bi.starschema.accounting
org.ofbiz.bi.starschema.order
org.ofbiz.bi.starschema.product
org.ofbiz.catalina.session
org.ofbiz.common.datasource
org.ofbiz.common.email
org.ofbiz.common.enum
org.ofbiz.common.geo
org.ofbiz.common.keyword
org.ofbiz.common.language
org.ofbiz.common.method
org.ofbiz.common.note
org.ofbiz.common.period
org.ofbiz.common.portal
org.ofbiz.common.property
org.ofbiz.common.status
org.ofbiz.common.theme
org.ofbiz.common.uom
org.ofbiz.common.user
org.ofbiz.content.compdoc
org.ofbiz.content.content
org.ofbiz.content.data
org.ofbiz.content.document
org.ofbiz.content.preference

Entity Reference Chart
1102 Total Entities

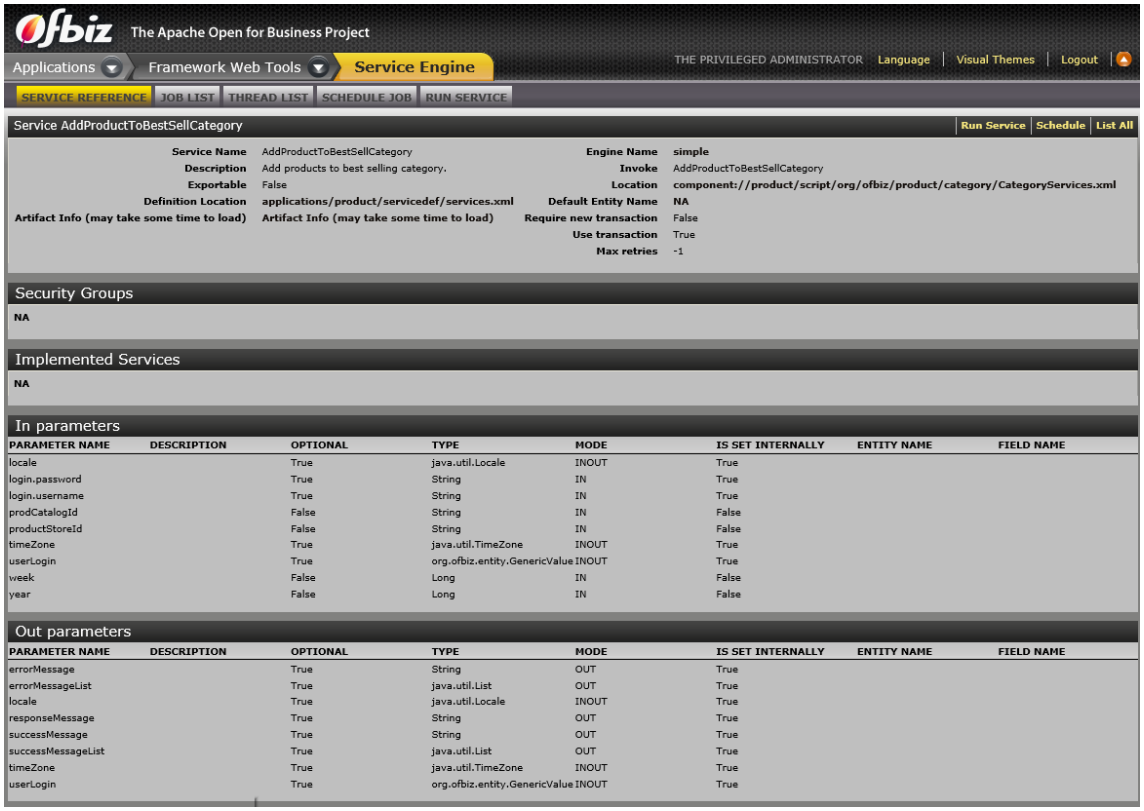
org.ofbiz.accounting.budget

ENTITY: Budget | TABLE: BUDGET
Budget Entity [View data] [Artifact Info (may take some time to load)]
file:/applications/accounting/entitydef/entitymodel.xml

Java Name	DB Name	Field Type	Java Type	SQL Type
budgetId Budget id	BUDGET_ID	id-ne	String	VARCHAR(20)
budgetTypeId Budget type id	BUDGET_TYPE_ID	id	String	VARCHAR(20)
customTimePeriodId Custom time period id	CUSTOM_TIME_PERIOD_ID	id	String	VARCHAR(20)
comments Comments	COMMENTS	comment	String	VARCHAR(255)
lastUpdatedStamp Last updated stamp	LAST_UPDATED_STAMP	date-time	java.sql.Timestamp	TIMESTAMP
lastUpdatedTxStamp Last updated tx stamp	LAST_UPDATED_TX_STAMP	date-time	java.sql.Timestamp	TIMESTAMP
createdStamp Created stamp	CREATED_STAMP	date-time	java.sql.Timestamp	TIMESTAMP
createdTxStamp Created tx stamp	CREATED_TX_STAMP	date-time	java.sql.Timestamp	TIMESTAMP

Relation	Relation Type
BudgetType FK Name: BUDGET_BGTTYP	one: 1) budgetTypeId : aa
CustomTimePeriod FK Name: BUDGET_CTP	one: 1) customTimePeriodId : aa
BudgetTypeAttr	many: 1) budgetTypeId : aa
BudgetAttribute	many: 1) budgetId : aa
BudgetItem	many: 1) budgetId : aa
BudgetReview	many: 1) budgetId : aa
BudgetRevision	many: 1) budgetId : aa

Figure 33 – OFBiz entity reference chart page



ofbiz The Apache Open for Business Project
Applications | Framework Web Tools | Service Engine
THE PRIVILEGED ADMINISTRATOR | Language | Visual Themes | Logout
SERVICE REFERENCE | JOB LIST | THREAD LIST | SCHEDULE JOB | RUN SERVICE

Service AddProductToBestSellCategory — Run Service | Schedule | List All

Service Name	AddProductToBestSellCategory	Engine Name	simple
Description	Add products to best selling category.	Invoke	AddProductToBestSellCategory
Exportable	False	Location	component://product/script/org/ofbiz/product/category/CategoryServices.xml
Definition Location	applications/product/servicedef/services.xml	Default Entity Name	NA
Artifact Info (may take some time to load)	Artifact Info (may take some time to load)	Require new transaction	False
		Use transaction	True
		Max retries	-1

Security Groups
NA

Implemented Services
NA

In parameters

PARAMETER NAME	DESCRIPTION	OPTIONAL	TYPE	MODE	IS SET INTERNALLY	ENTITY NAME	FIELD NAME
locale		True	java.util.Locale	INOUT	True		
login.password		True	String	IN	True		
login.username		True	String	IN	True		
prodCatalogId		False	String	IN	False		
productStoreId		False	String	IN	False		
timeZone		True	java.util.TimeZone	INOUT	True		
userLogin		True	org.ofbiz.entity.GenericValue	INOUT	True		
week		False	Long	IN	False		
year		False	Long	IN	False		

Out parameters

PARAMETER NAME	DESCRIPTION	OPTIONAL	TYPE	MODE	IS SET INTERNALLY	ENTITY NAME	FIELD NAME
errorMessage		True	String	OUT	True		
errorMessageList		True	java.util.List	OUT	True		
locale		True	java.util.Locale	INOUT	True		
responseMessage		True	String	OUT	True		
successMessage		True	String	OUT	True		
successMessageList		True	java.util.List	OUT	True		
timeZone		True	java.util.TimeZone	INOUT	True		
userLogin		True	org.ofbiz.entity.GenericValue	INOUT	True		

Figure 34 – OFBiz service reference page

9.3 Label manager

The label manager allows users to search for a given application text and list missing translations.

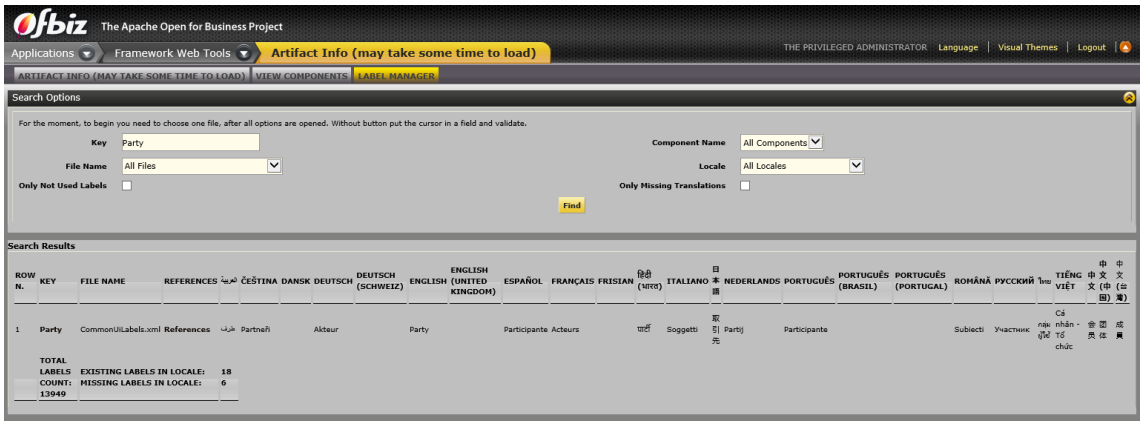


Figure 35 – OFBiz label management page

9.4 Entity engine tools

The entity engine tools allow system administrators to search for entities, view its relations with other entities, create new records for a given entity or search a specific record.

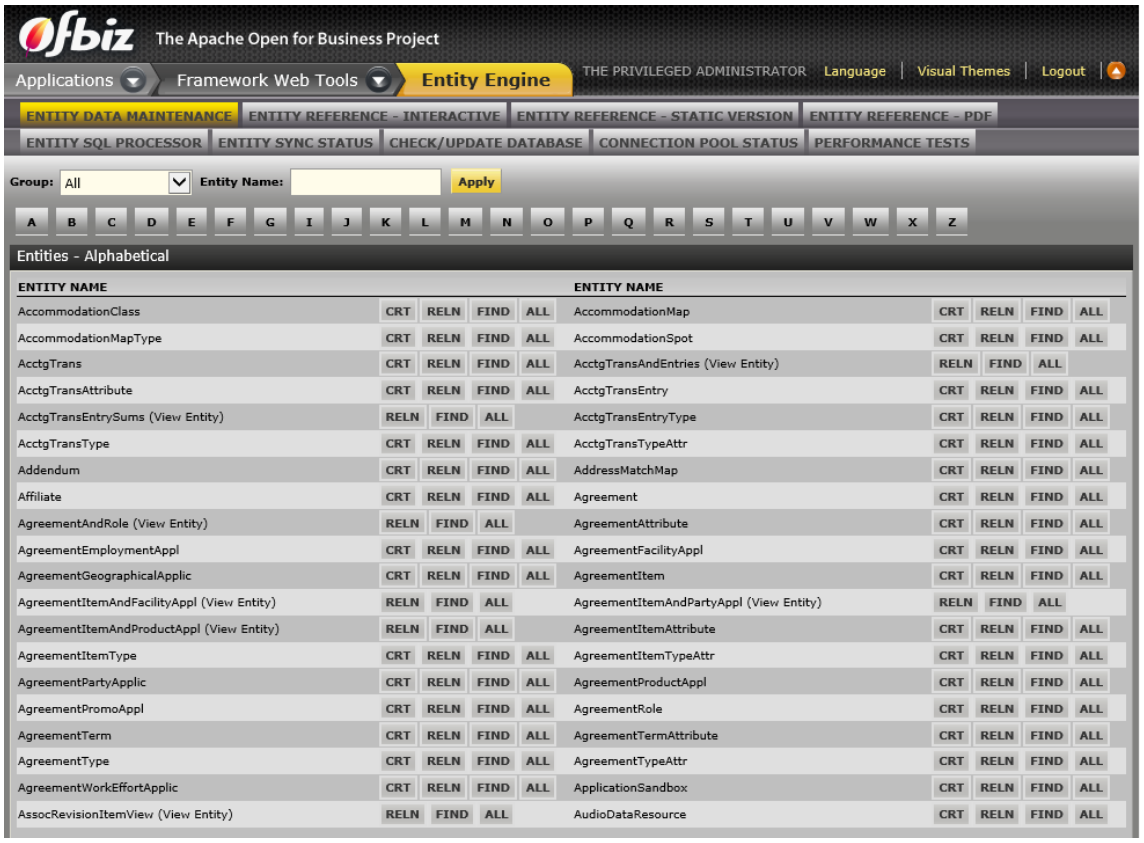


Figure 36 – OFBiz entity data maintenance page

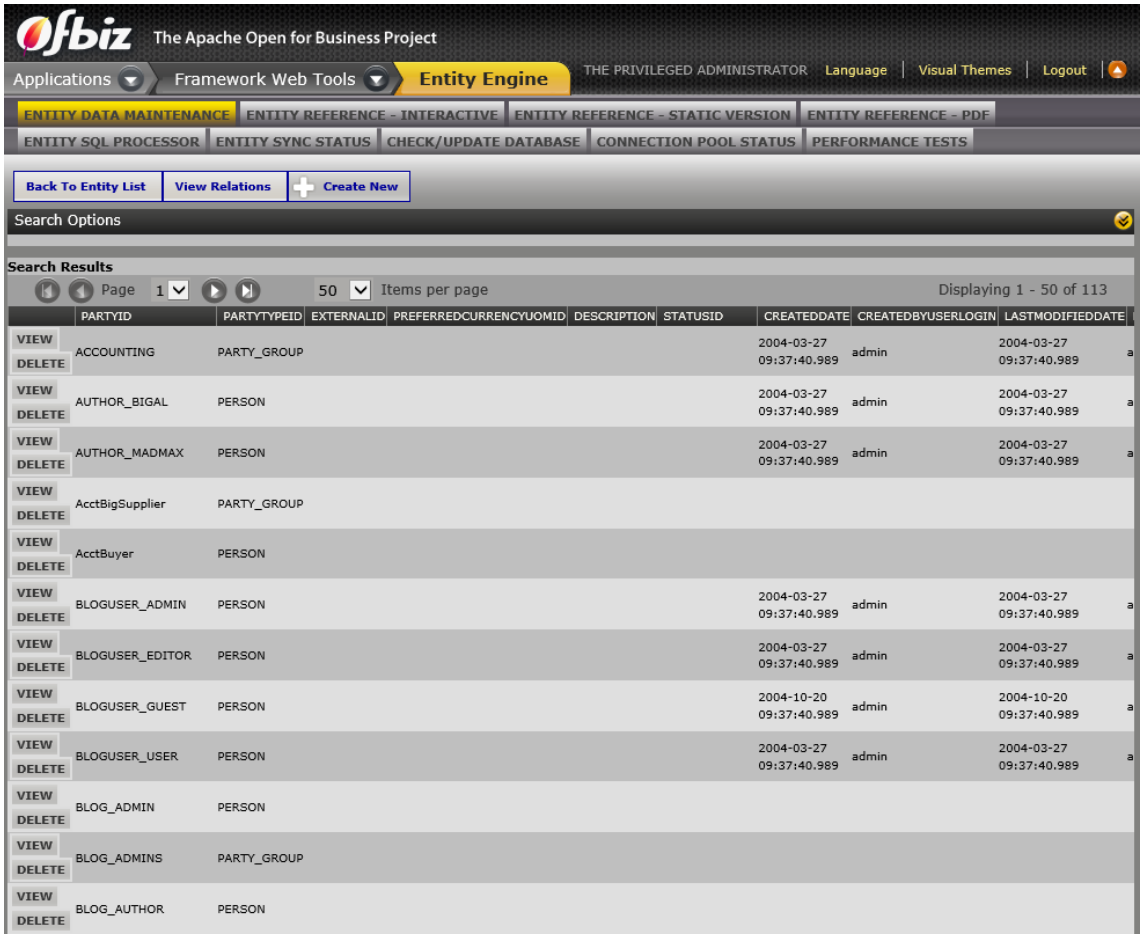


Figure 37 – OFBiz entity values search page

9.5 Entity XML tools

The entity xml tools provide the capacity to export records from the database to XML to be re-imported later into another OFBiz instance.

9.6 Service engine

The service engine tools provide the service reference that is also available in the general artifact information tools and allows services to be executed manually, providing values for the input parameter of the service and checking its output values. Users are also able to schedule a job to be executed only once or periodically as well as to check the list of all jobs that were executed or are scheduled.

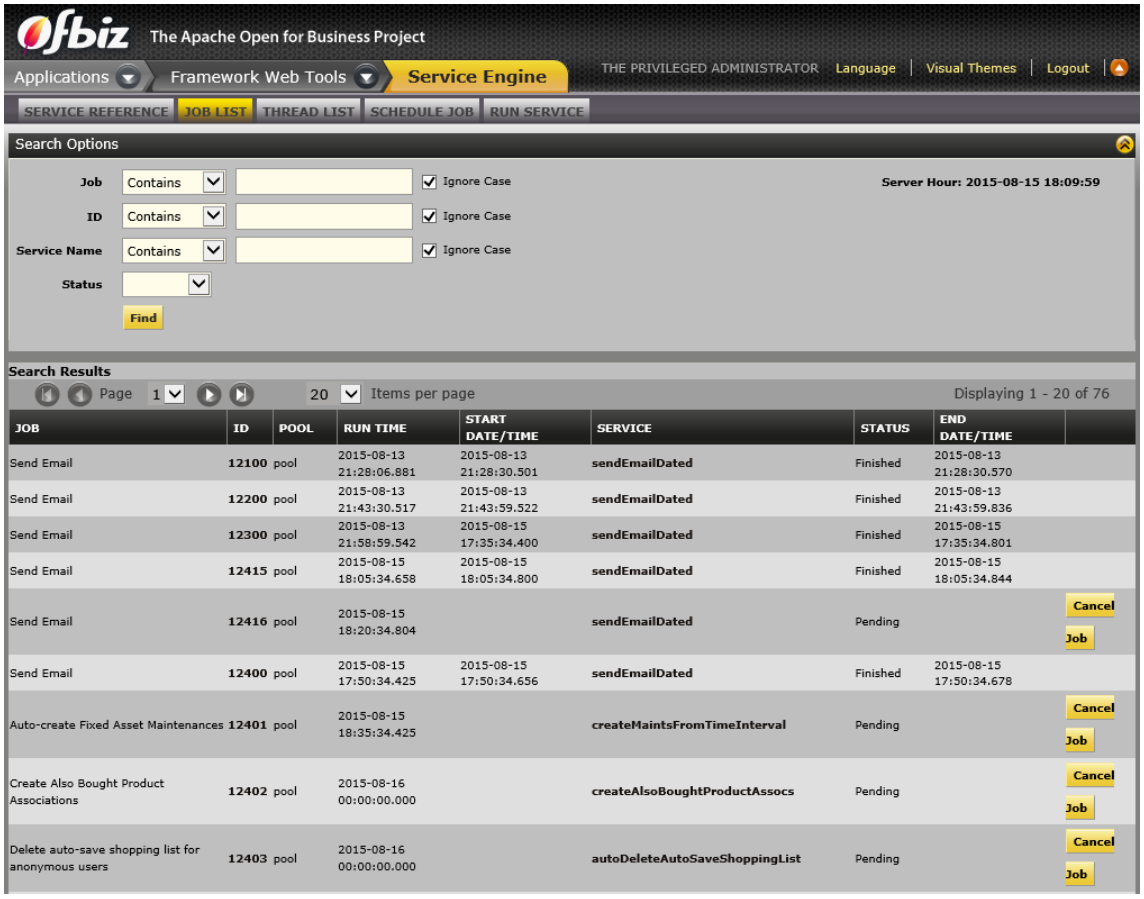


Figure 38 – OFBiz job search page

9.7 Performance tests

Provides a single page that executes a predefined set of calls to measure the performance of the OFBiz database.

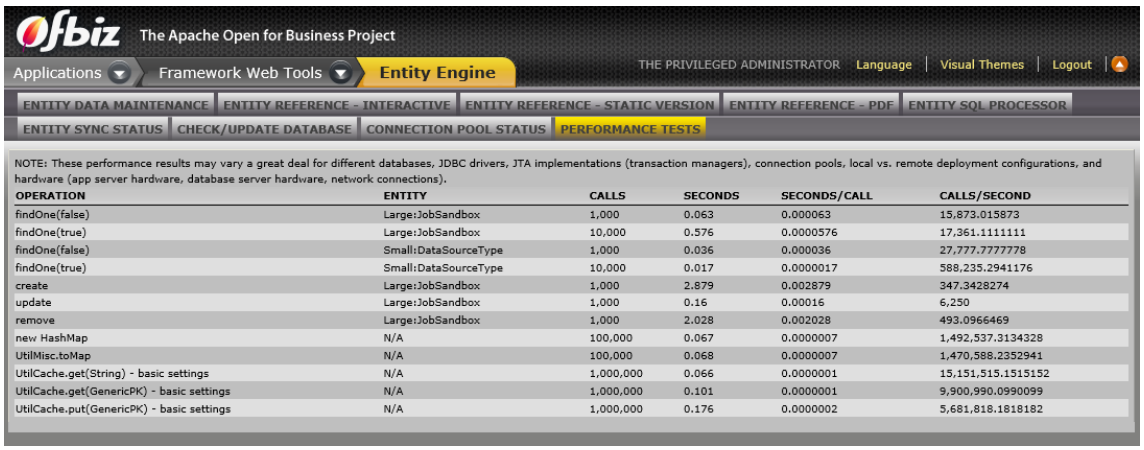


Figure 39 – OFBiz performance tests page

10 References

The Apache Software Foundation. (08 de Oct de 2015). *Apache OFBiz™*. Retrieved from https://ofbiz.apache.org/

The Apache Software Foundation. (08 de Oct de 2015). *OFBiz Project Open Wiki*. Retrieved from https://cwiki.apache.org/confluence/display/OFBIZ/Home

The Apache Software Foundation. (08 de Oct de 2015). *OFBiz Technical Documentation*. Retrieved from https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Technical+Documentation+-+Home+Page

Appendix II

Instituto Superior de Engenharia do Porto

SAF-T PT Report
Alpha Tests

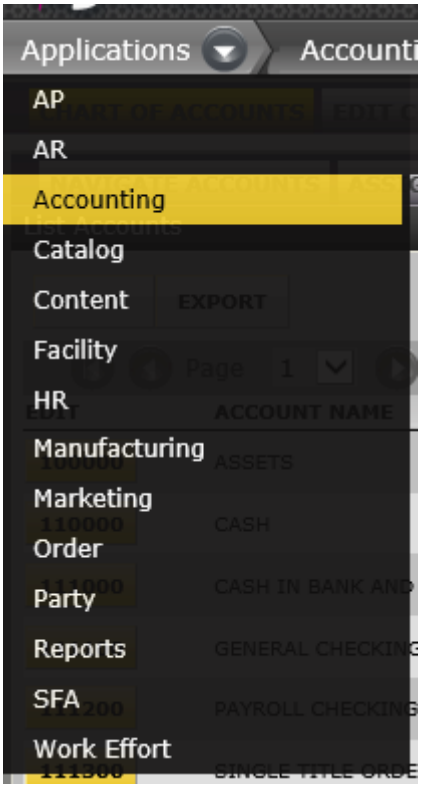
Jorge Almeida
05-10-2015

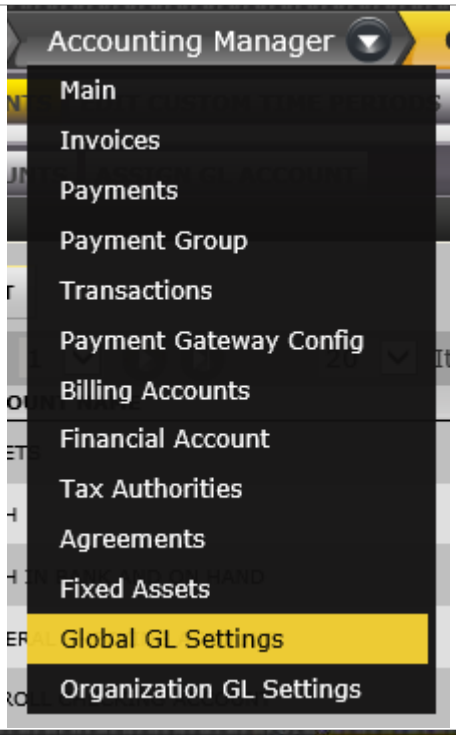
Table of contents

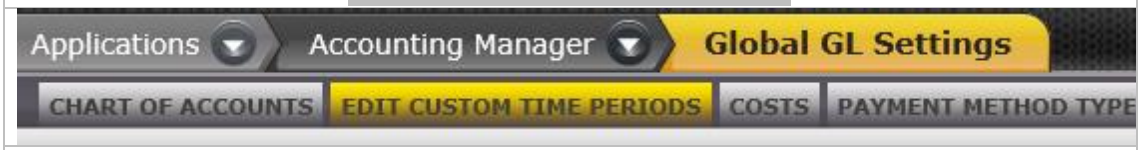
Custom Time Period (equal dates)

Test Id	1
Purpose of the test	Ensure that an error message is displayed to the user and the download of the report is unavailable when the selected Custom Time Period has equal start and end dates.
Test steps	Create or edit a custom time period; Generate the SAF-T PT report with the Custom Time Period created/edited above.
Expected result	An error message should be displayed saying that the Custom Time Period has equal start and end dates. The download button should not be visible.

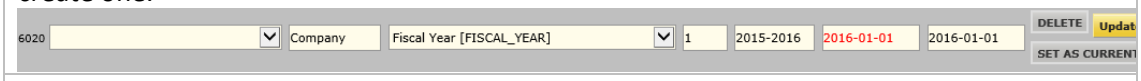
Go to the Accounting module, access the Global GL Settings and then click on the “Edit custom time period” tab.



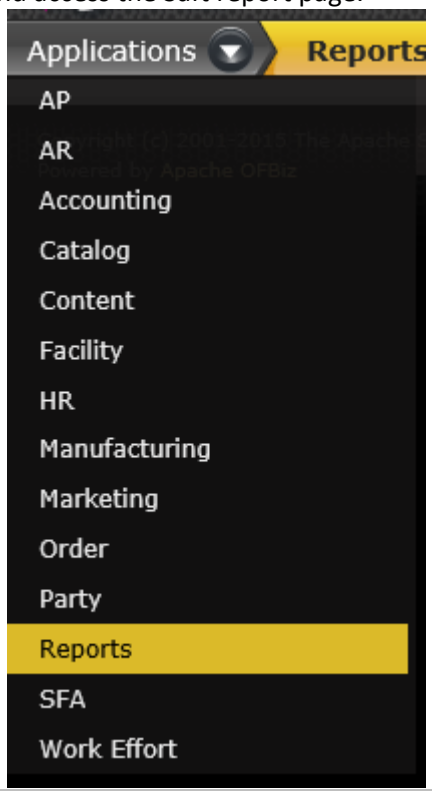


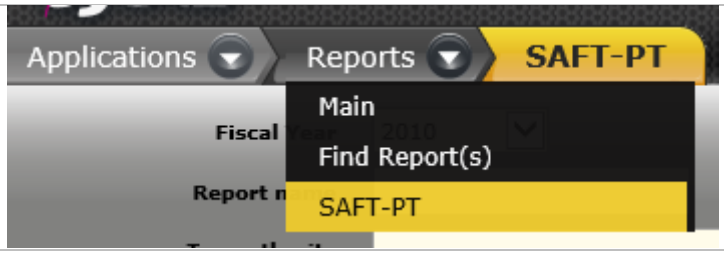


Change an existing time period to have equal start and end dates. If one does not exist create one.



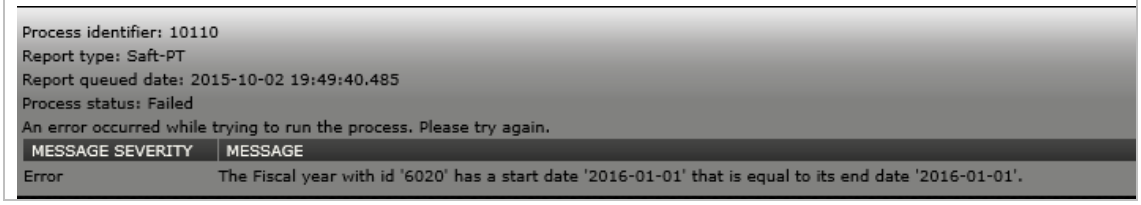
Go to the Reports module and access the Saft report page.





Select the fiscal year created/changed above and click “Submit”

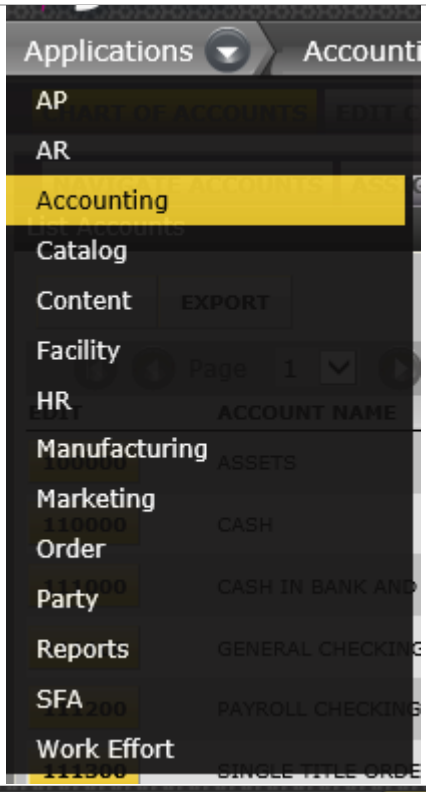
The following message should be visible and the download of the report should not be allowed.

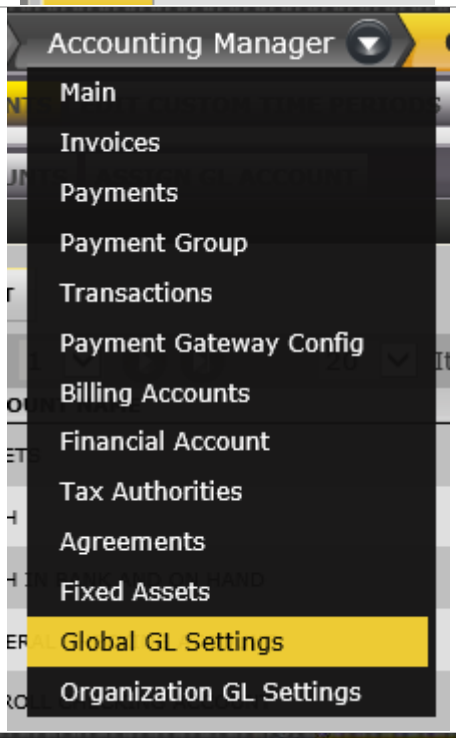


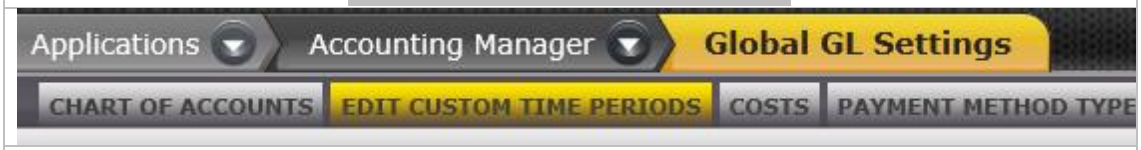
Custom Time Period (end before start)

Test Id	2
Purpose of the test	Ensure that an error message is displayed to the user and the download of the report is unavailable when the selected Custom Time Period ends before it starts.
Test steps	Create or edit a Custom Time Period; Generate the SAF-T PT report with the Custom Time Period created/edited above.
Expected result	An error message should be displayed saying that the selected Custom Time Period ends before it starts. The download button should not be visible.

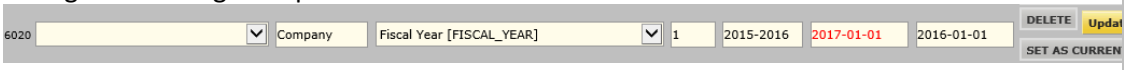
Go to the Accounting module, access the Global GL Settings and then click on the “Edit custom time period” tab.



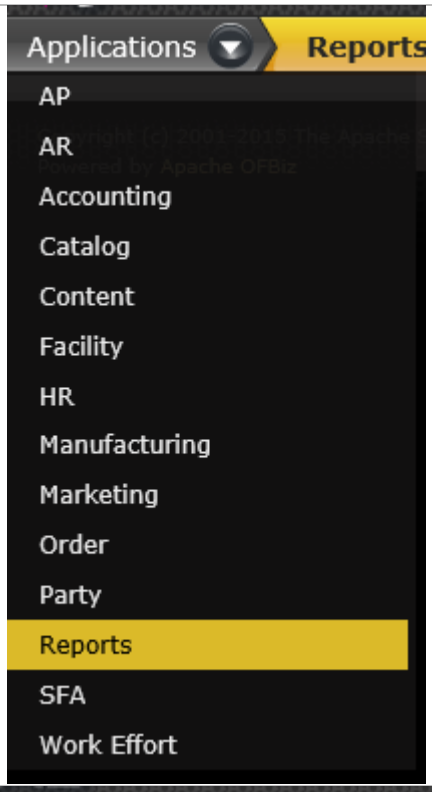


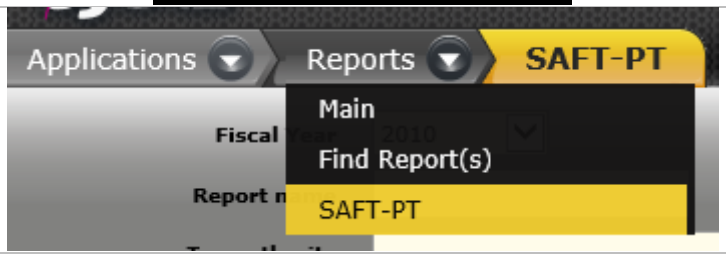


Change an existing time period to end before it starts. If one does not exist create one.



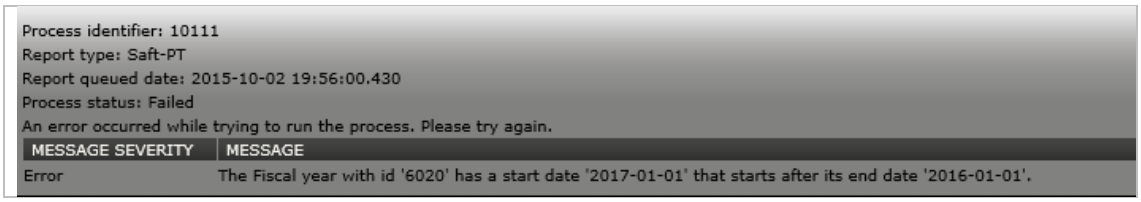
Go to the Reports module and access the Saft report page.





Select the fiscal year created/changed above and click “Submit”

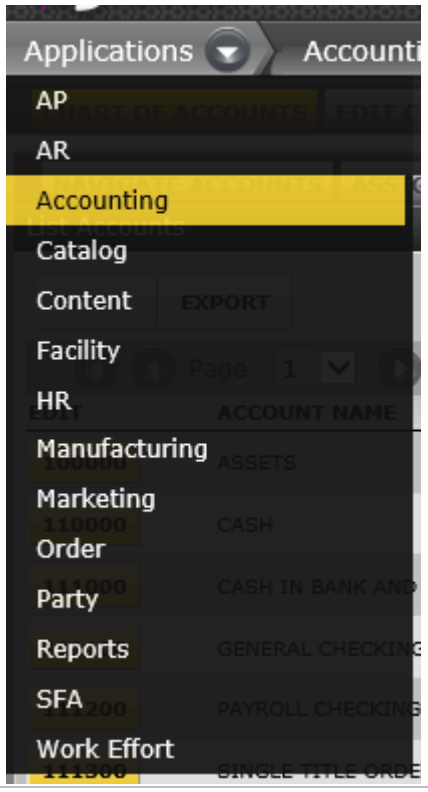
The following message should be visible and the download of the report should not be allowed.

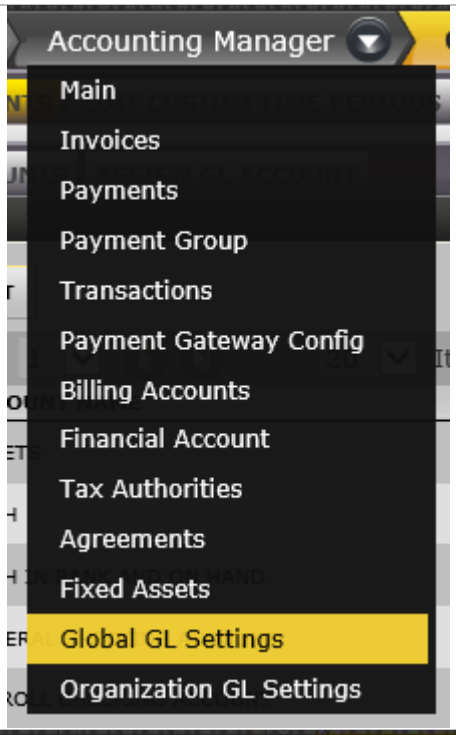


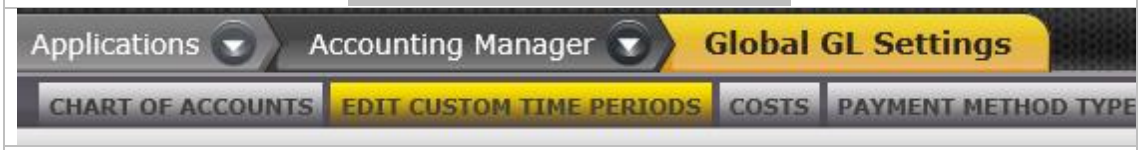
Party without address

Test Id	3
Purpose of the test	Ensure that an error message is presented to the user and the download of the report is unavailable when a Party has no address for the specified fiscal year.
Test steps	Create or edit a Custom Time Period; Expire the address for an existing Party; Generate the SAF-T PT report with the Custom Time Period created/edited above.
Expected result	An error message should be displayed saying that a Party has no address specified. The download button should not be visible.

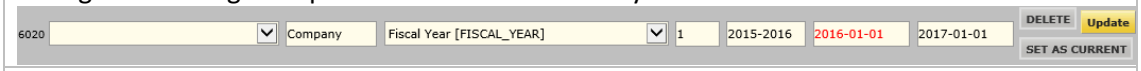
Go to the Accounting module, access the Global GL Settings and then click on the “Edit custom time period” tab.



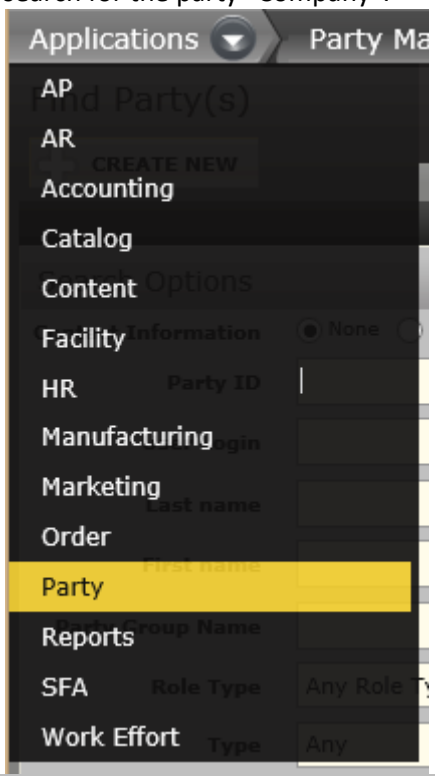


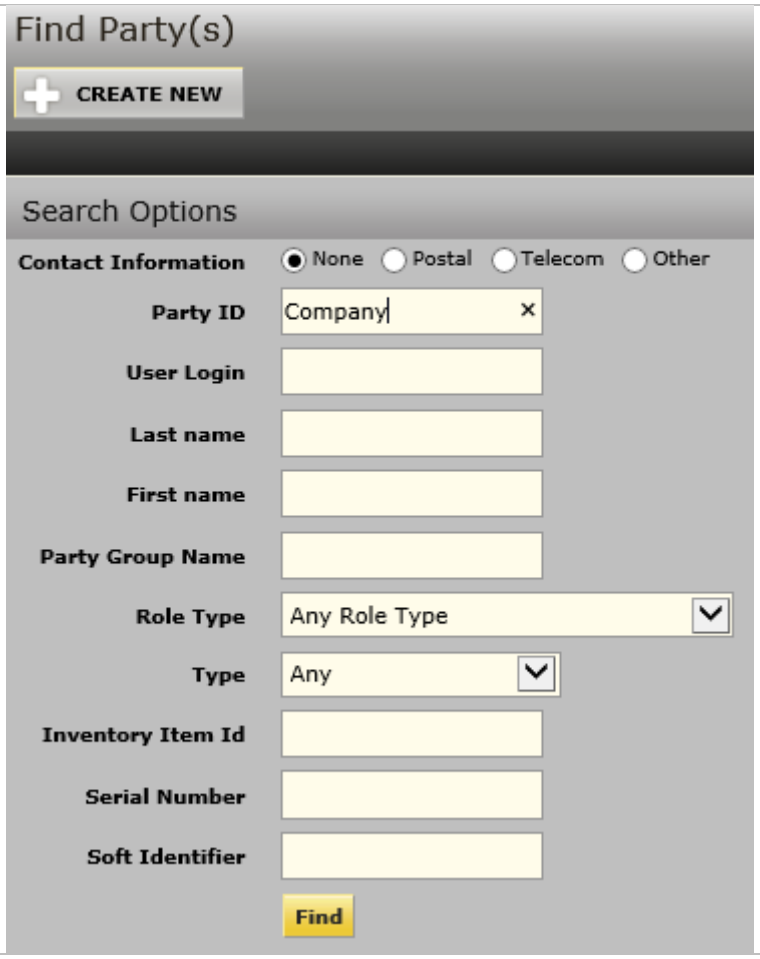


Change an existing time period to start on the next year.

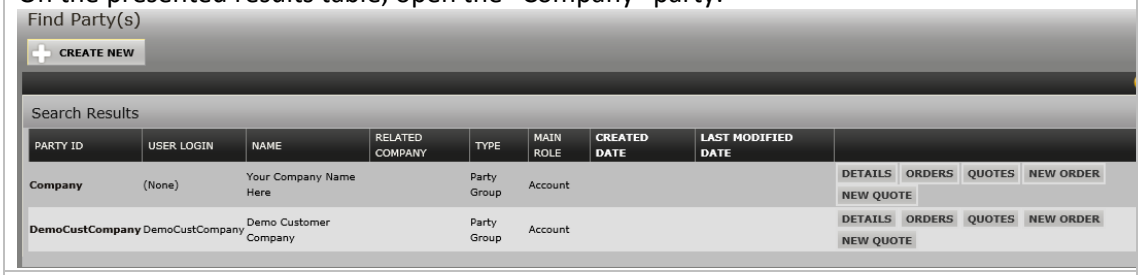


Go to the Party module and search for the party “Company”.

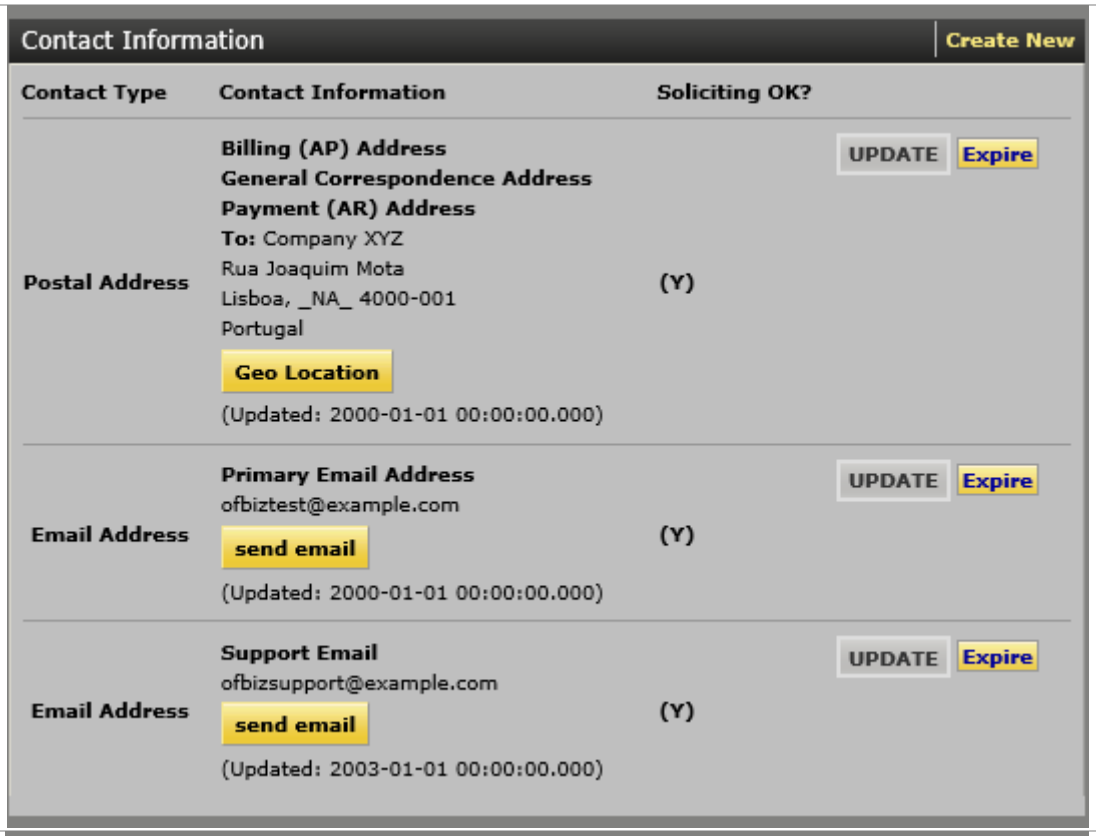


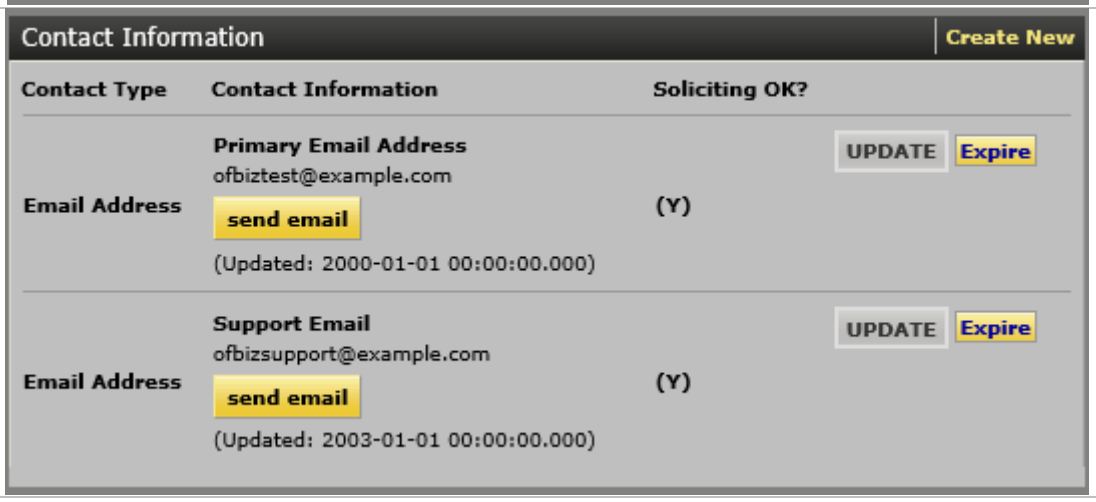


On the presented results table, open the “Company” party.

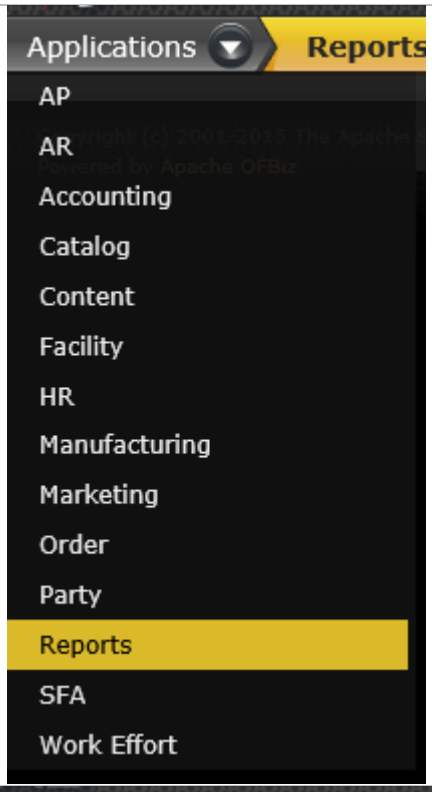


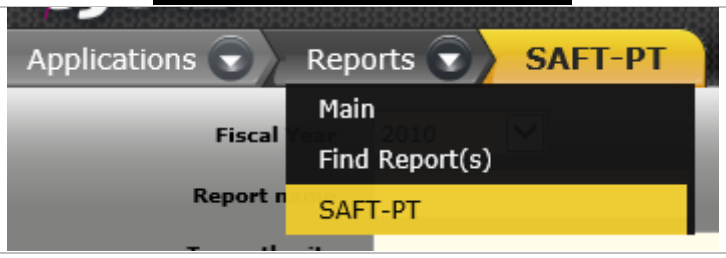
On the contacts panel, click on the “Expire” button for the postal address.





Go to the Reports module and access the Saft report page.





Select the fiscal year created/changed above and click “Submit”

The following message should be visible and the download of the report should not be allowed.

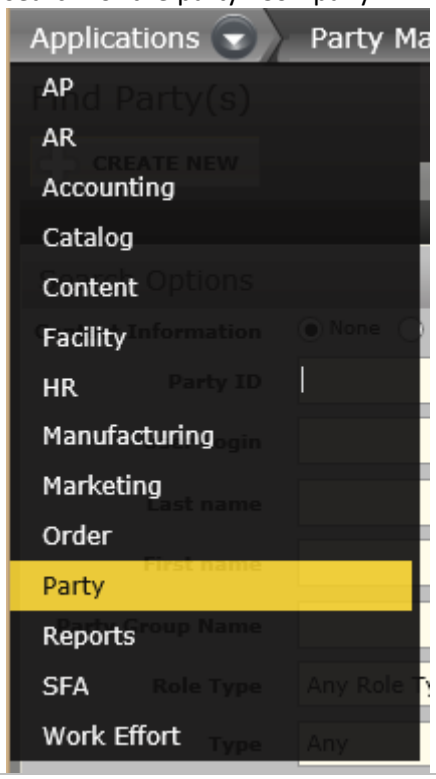
Process identifier: 10112
Report type: Saft-PT
Report queued date: 2015-10-02 19:58:30.211
Process status: Failed
An error occurred while trying to run the process. Please try again.

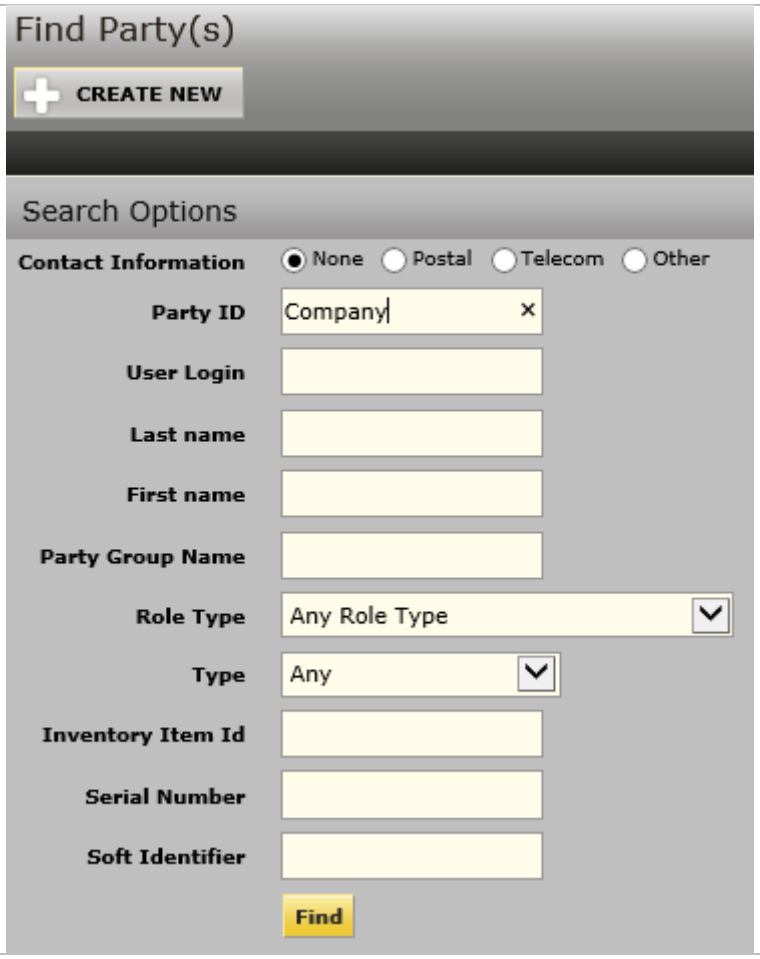
MESSAGE SEVERITY	MESSAGE
Warning	No active postal address found for party with id 'Company', contact mechanism purpose id '' and country geo id 'PRT' within the specified fiscal year. If any postal address is specified for the party it will be used.
Error	No active postal address found for party with id 'Company' and country geo id 'PRT' within the specifie fiscal year.

Party with more than one postal address

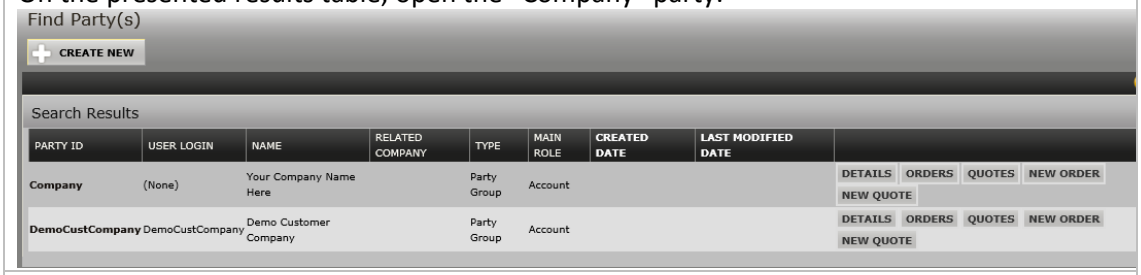
Test Id	4
Purpose of the test	Ensure that a warning message is presented to the user when a Party has more than one postal address for the specified fiscal year.
Test steps	Create or edit a given Party to have more than one Postal Address; Generate the SAF-T PT report.
Expected result	A warning message should be displayed saying that a Party has more than one Postal Address. The Postal Address that was created or edited more recently is the one that was exported. The report should be valid according to the rules of the Report Validation Tool.

Go to the Party module and search for the party “Company”.

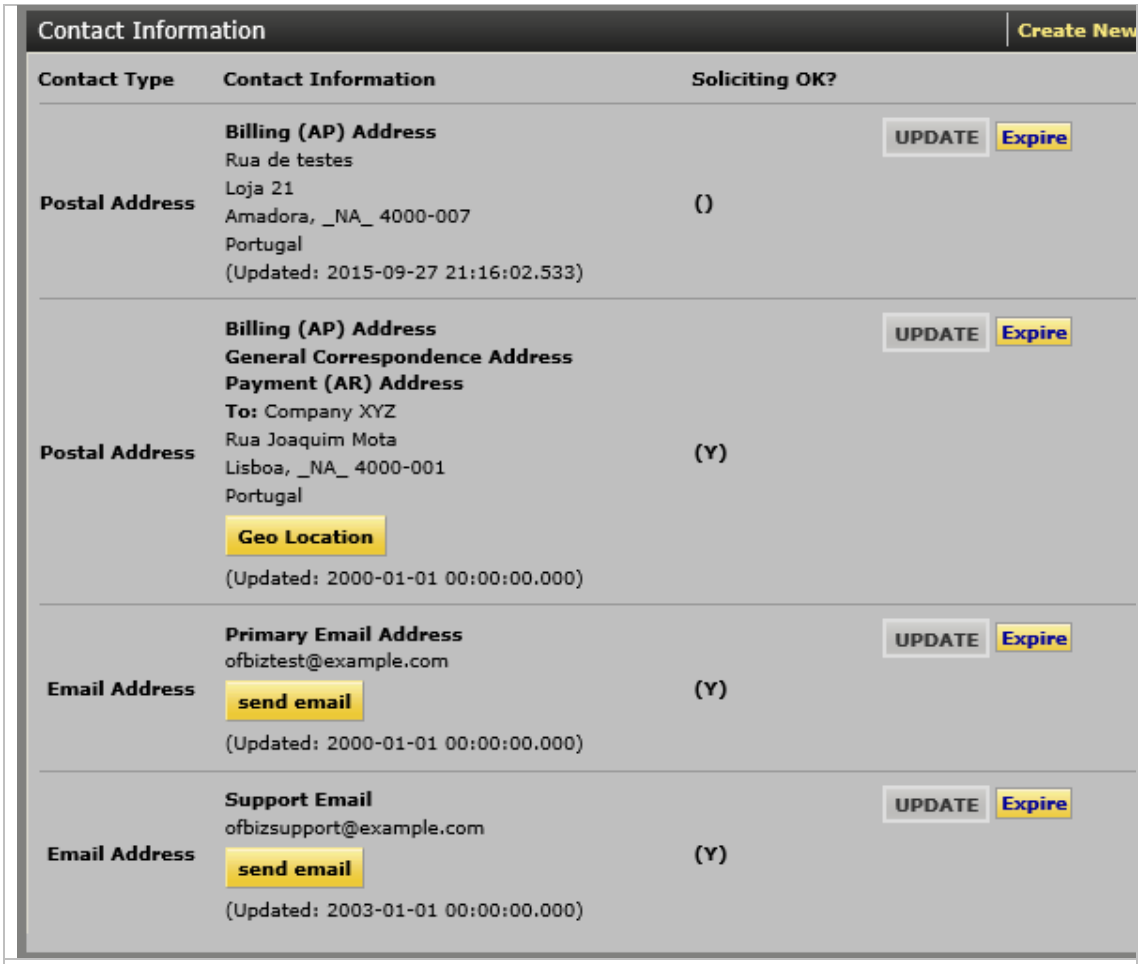




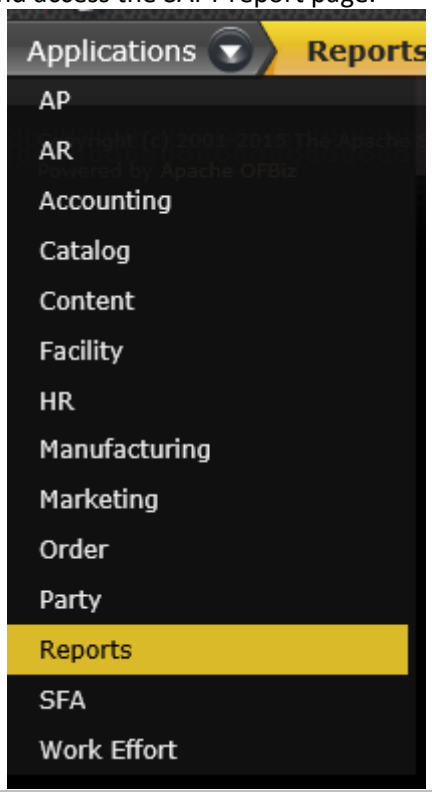
On the presented results table, open the “Company” party.

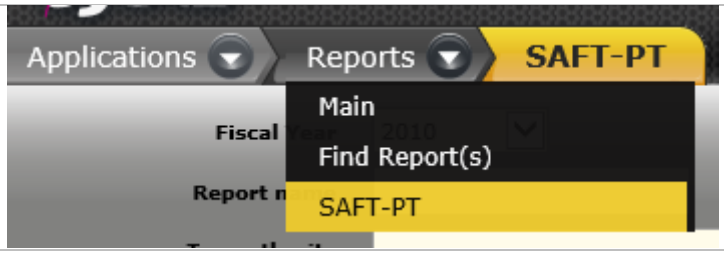


Add a new Postal Address and assign the “Billing Location” role to it so that the party has two distinct billing location postal addresses:



Go to the Reports module and access the SAFT report page.





Select the fiscal year for the current year and click “Submit”.

The following warning message should be visible:

Download the report and check that the Address for the company is the one that was created above:

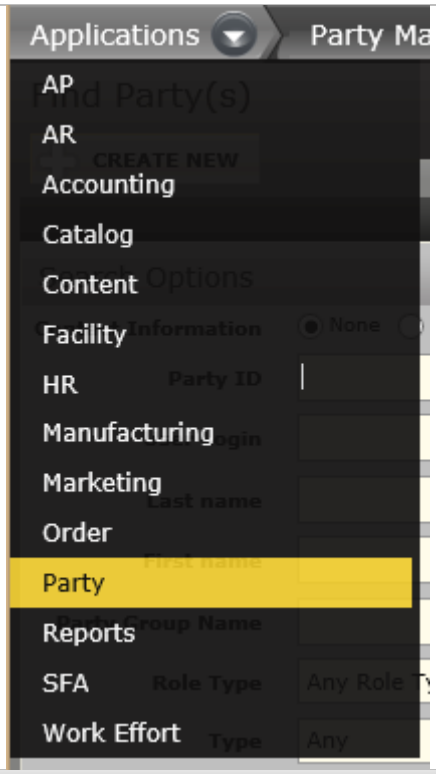
```
<AuditFile xmlns="urn:OECD:StandardAuditFile-Tax:PT_1.03_01">
    <Header>
        <AuditFileVersion>1.03_01</AuditFileVersion>
        <CompanyID>510026311</CompanyID>
        <TaxRegistrationNumber>510026311</TaxRegistrationNumber>
        <TaxAccountingBasis>F</TaxAccountingBasis>
        <CompanyName>Your Company Name Here</CompanyName>
        <CompanyAddress>
            <AddressDetail>Rua de testes, Loja 21, Amadora, 4000-007</AddressDetail>
            <City>Amadora</City>
            <PostalCode>4000-007</PostalCode>
            <Country>PT</Country>
        </CompanyAddress>
```

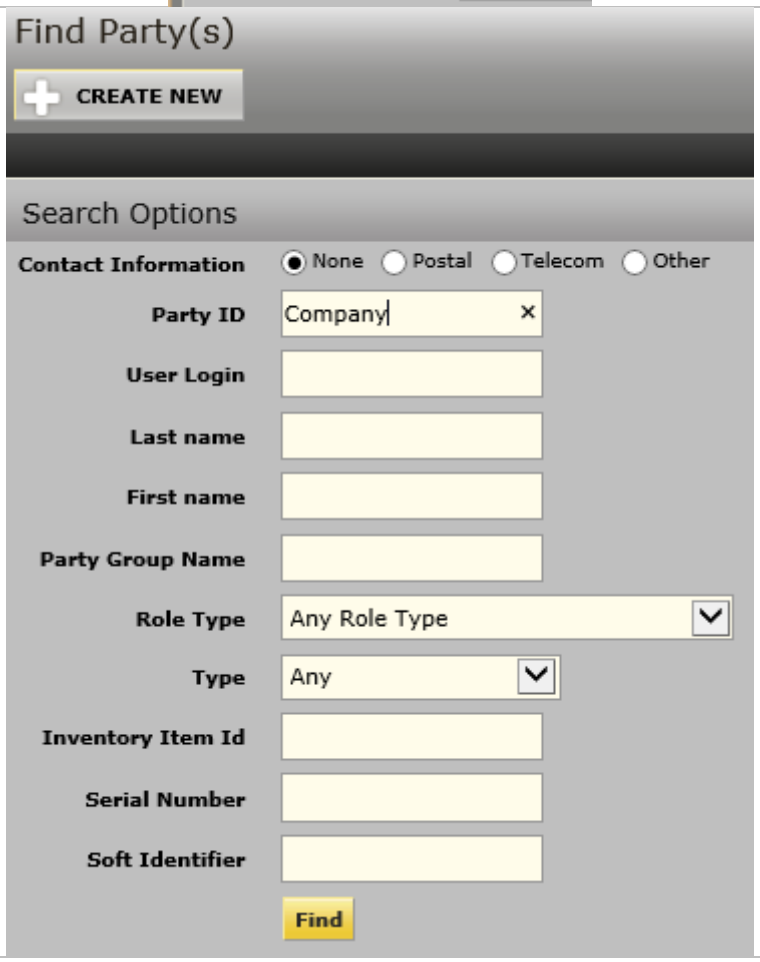
Check that the report is valid using the SAFT validation tool:

Postal address without Region

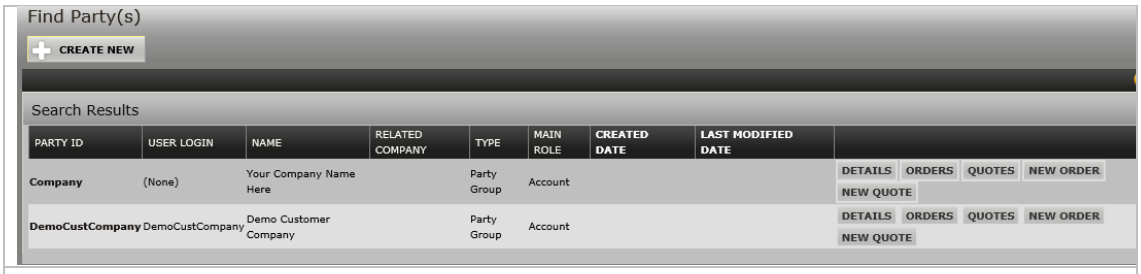
Test Id	5
Purpose of the test	Ensure that no Region is displayed on the report when the Postal Address of a given Party doesn't have a Region specified.
Test steps	Edit the postal address of a given Party and change the value of the Region to "_NA_" or create a new postal address with the same value for the Region field. Generate the SAF-T PT report.
Expected result	No Region should appear on the report; The report should be valid according to the rules of the Report Validation Tool.

Go to the Party module and search for the party "Company".

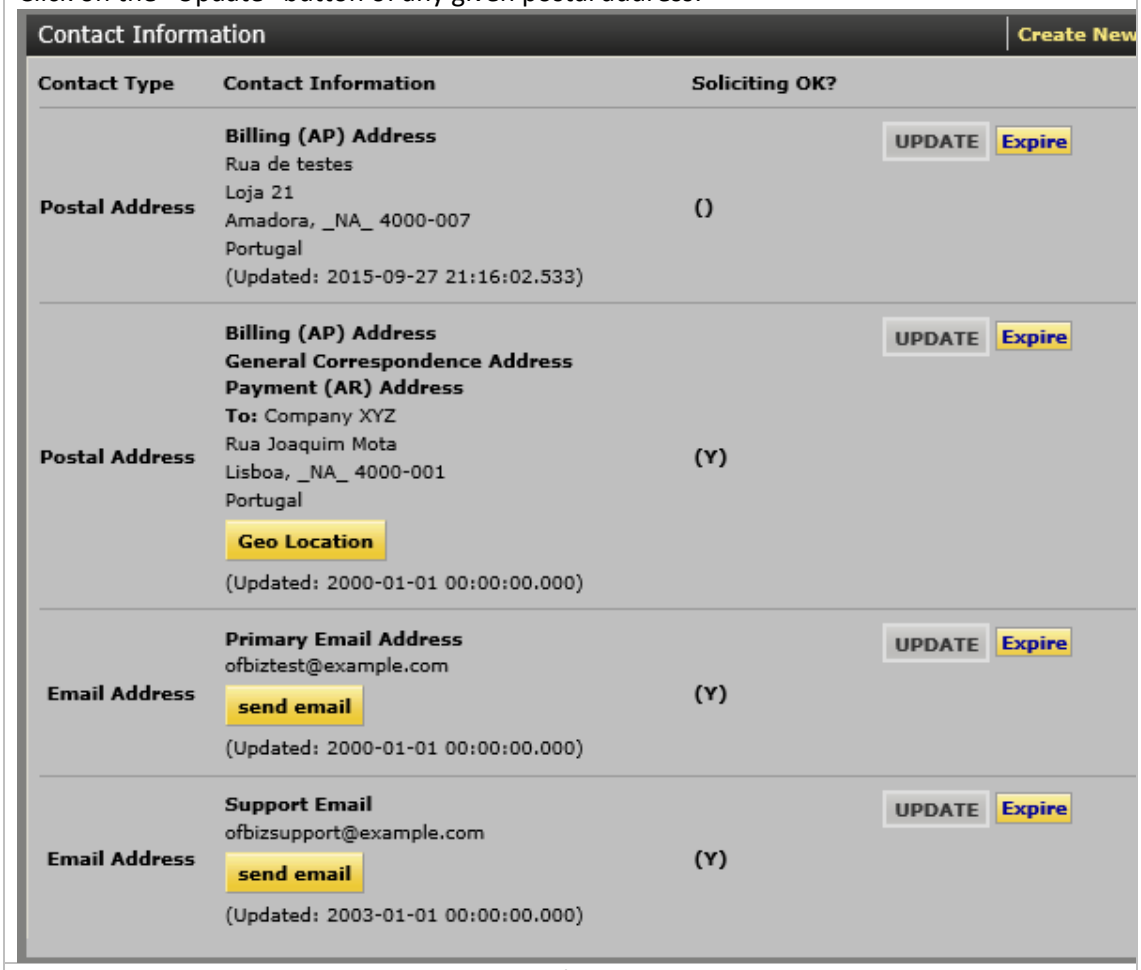




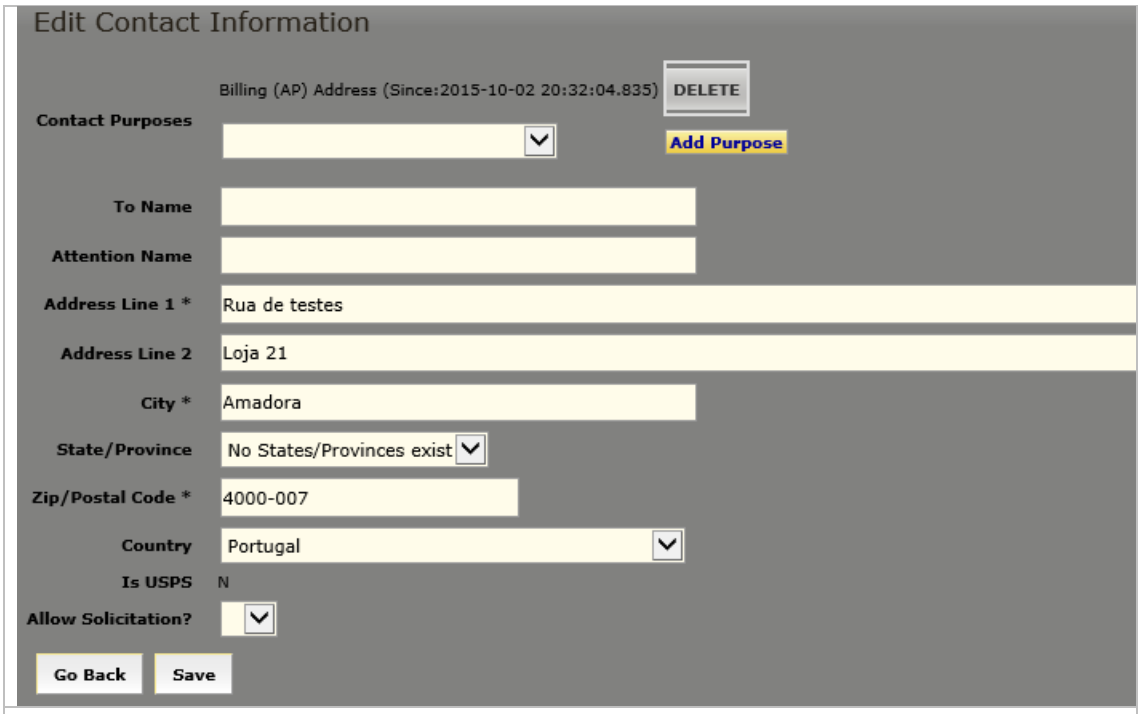
On the presented results table, open the “Company” party.



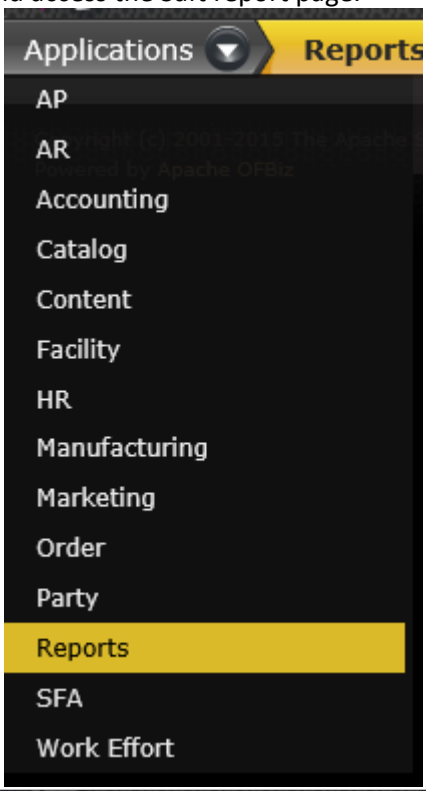
Click on the “Update” button of any given postal address:

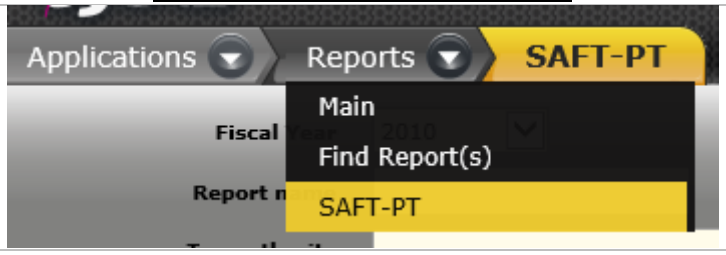


Make sure that the region has the “No States/Provinces exist” value:



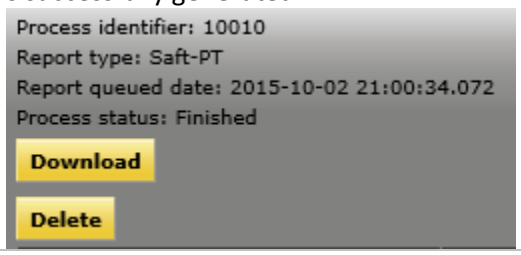
Go to the Reports module and access the Saft report page.





Select the fiscal year for the current year and click “Submit”.

Check that the report was successfully generated:



Download the report and check that no region is specified for the party used above:

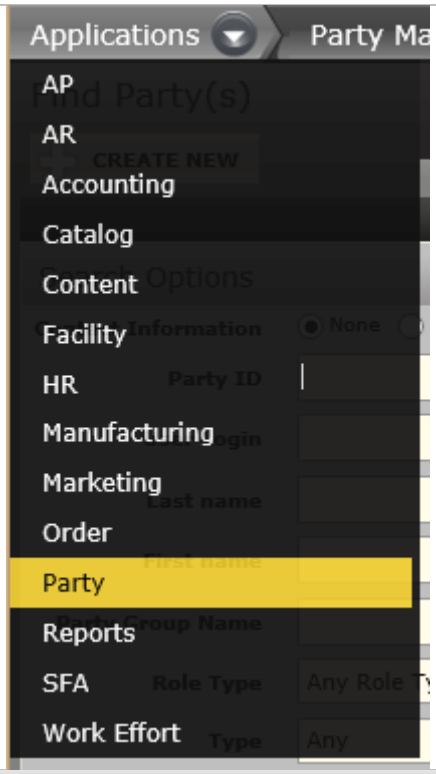
```
<CompanyAddress>
    <AddressDetail>Rua de testes, Loja 21, Amadora, 4000-007</AddressDetail>
    <City>Amadora</City>
    <PostalCode>4000-007</PostalCode>
    <Country>PT</Country>
</CompanyAddress>
```

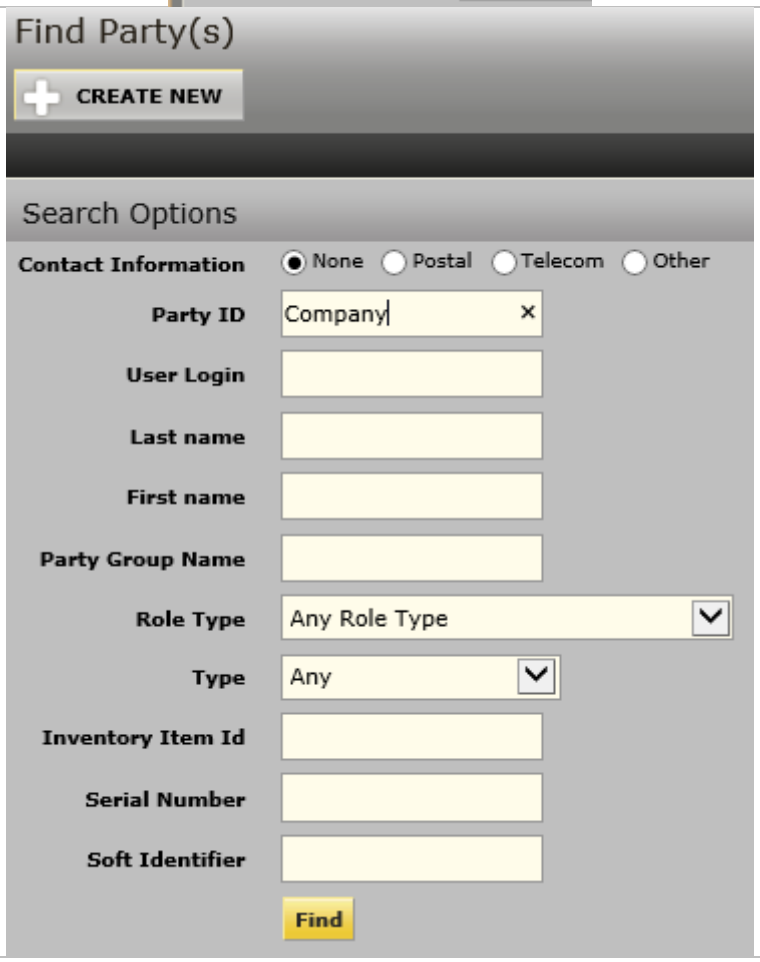
Check that the report is valid using the SAFT validation tool:

Company Party with invalid postal code

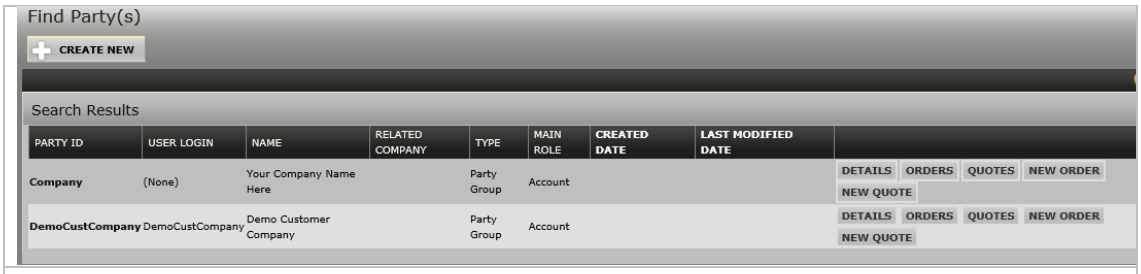
Test Id	6
Purpose of the test	Ensure that a error message is displayed to the user when the Postal Code of the Company Party is invalid according to the Portuguese Postal Code structure.
Test steps	Edit the Company Party; Change the postal code of the postal address to be invalid according to the Portuguese postal code structure; Generate the SAF-T PT report.
Expected result	An error message should be displayed saying that the Company Party contains an invalid postal code. No download button should be visible.

Go to the Party module and search for the party “Company”.

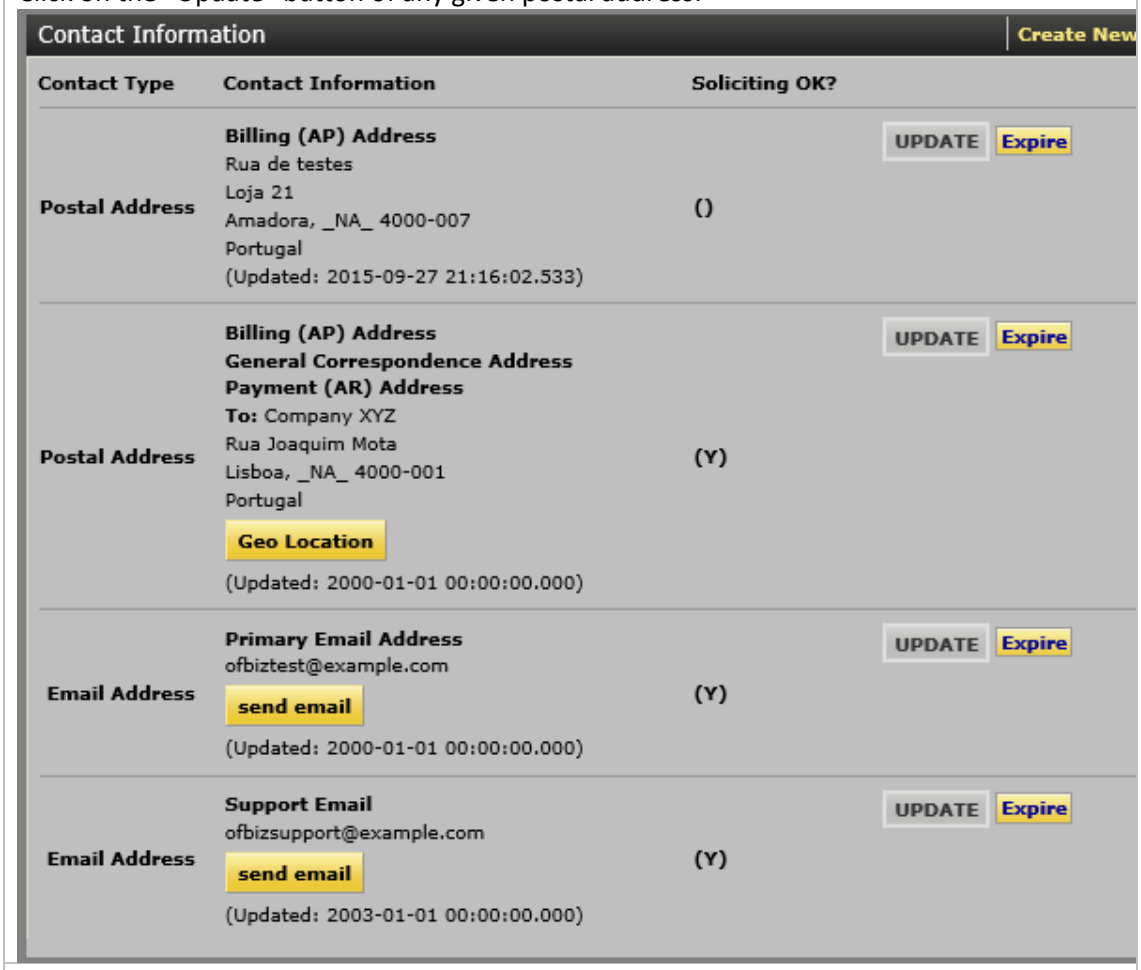




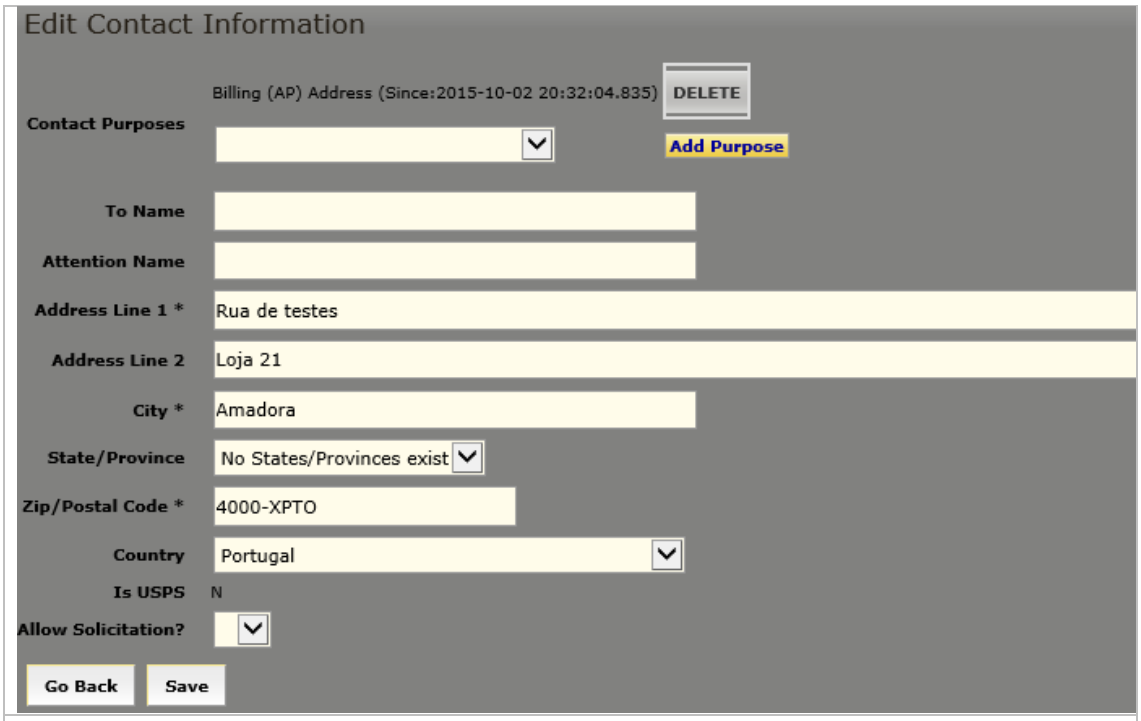
On the presented results table, open the “Company” party.



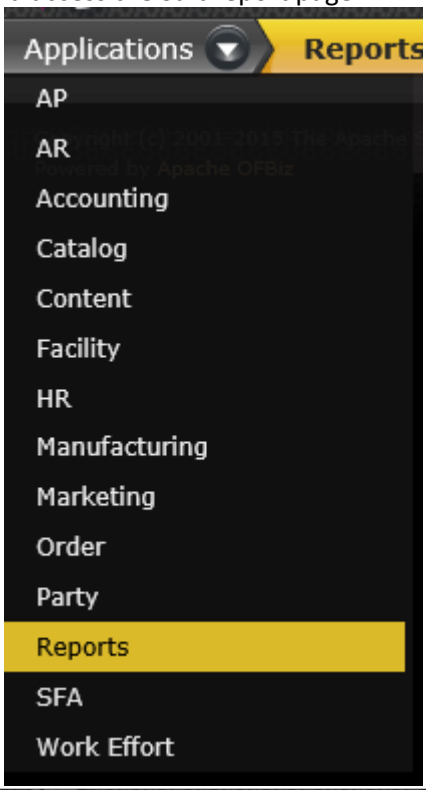
Click on the “Update” button of any given postal address:

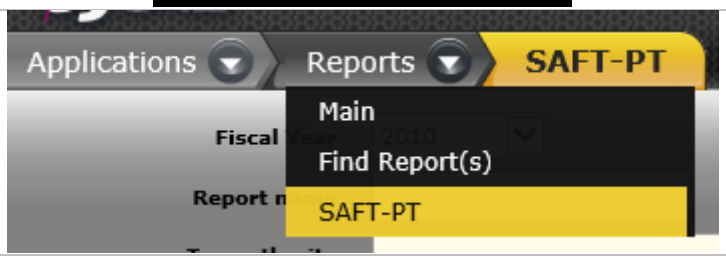


Change the postal address to a wrongly formatted Portuguese postal address:



Go to the Reports module and access the Saft report page.





Select the fiscal year for the current year and click “Submit”.

Check that the report generation failed with the expected error:

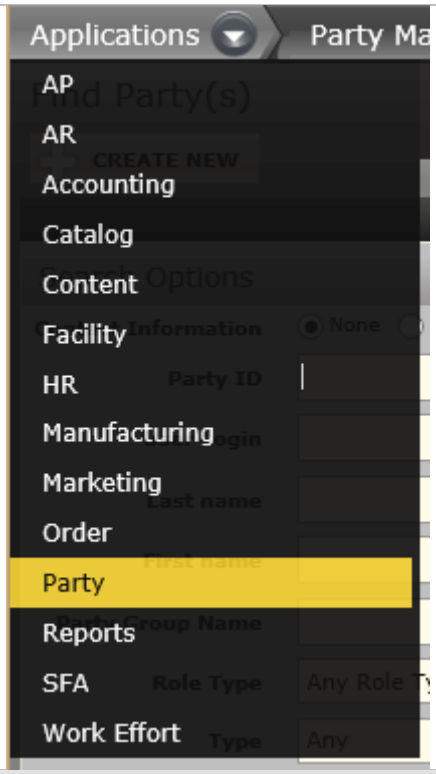
Process identifier: 10021
Report type: Saft-PT
Report queued date: 2015-10-02 21:09:46.549
Process status: Failed
An error occurred while trying to run the process. Please try again.

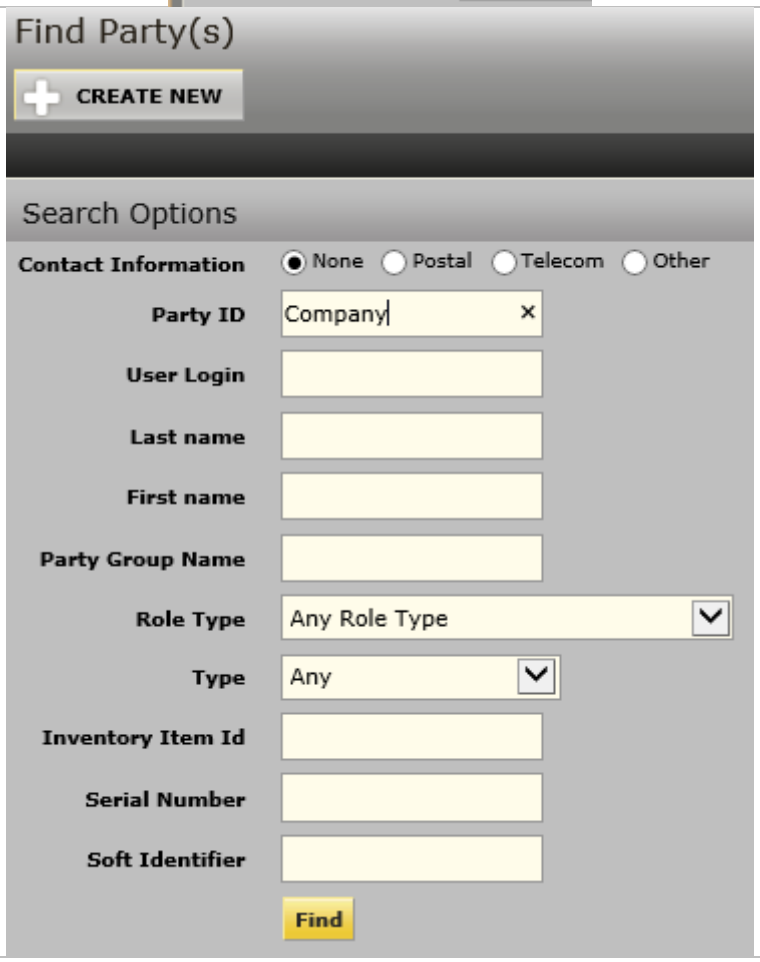
MESSAGE SEVERITY	MESSAGE
Warning	More than one active postal address found for party with id 'Company' and country geo id 'PRT' within the specified fiscal year. The most recently updated one will be used.
Error	Postal Code '4000-XPTO' for party 'Company' is invalid.

Company party without tax information

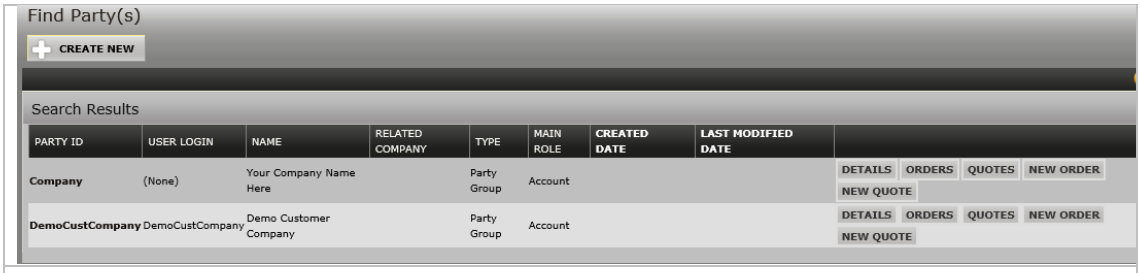
Test Id	7
Purpose of the test	Ensure that an error message is presented to the user when the Company Party has no Tax information.
Test steps	Edit the Company Party; Delete the Tax Information for Portugal; Generate the SAF-T PT report.
Expected result	An error message should be displayed saying that the Company Party doesn't have Tax Information for Portugal. No download button should be visible.

Go to the Party module and search for the party "Company".



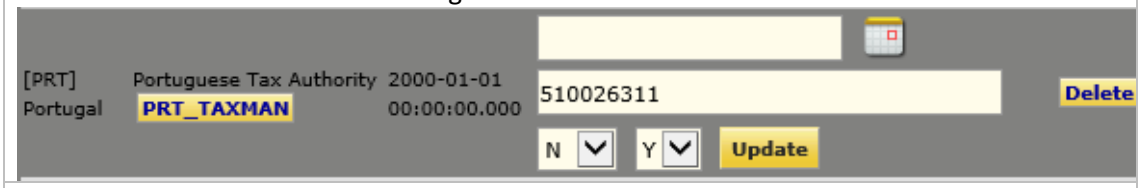


On the presented results table, open the “Company” party.

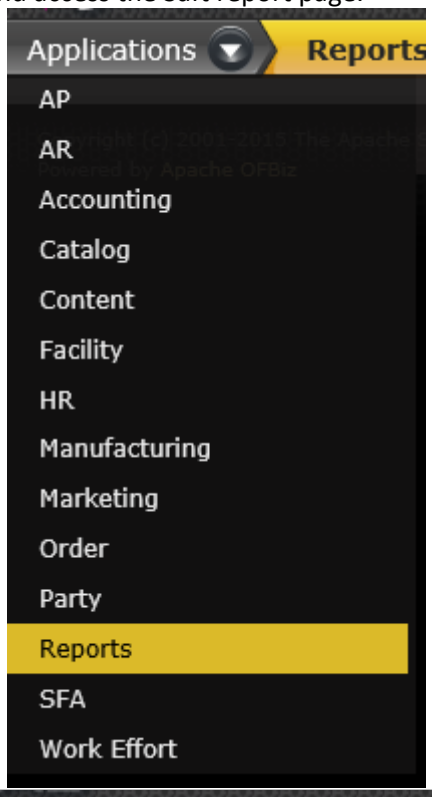


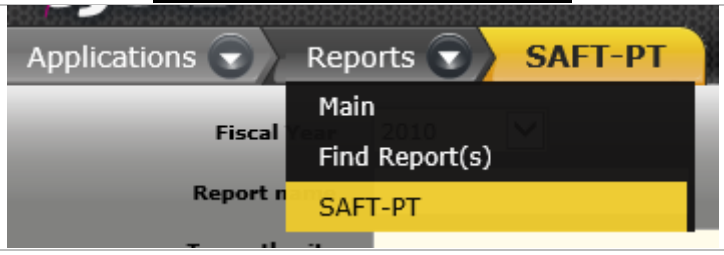
Go to the “Tax Infos” tab:

Delete the tax information for Portugal:



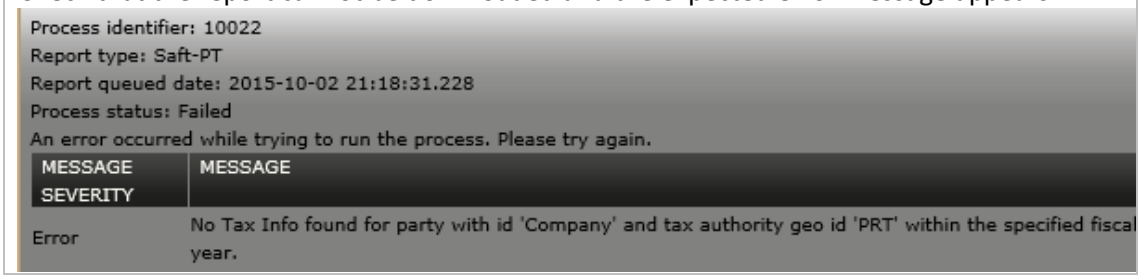
Go to the Reports module and access the Saft report page.





Select the fiscal year for the current year and click “Submit”.

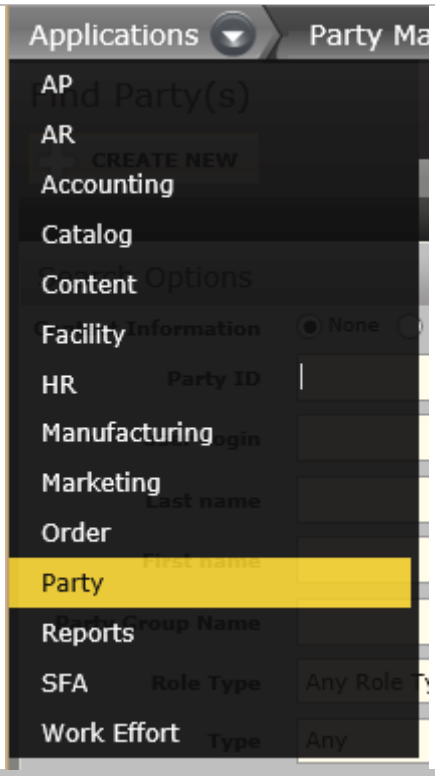
Check that the report cannot be downloaded and the expected error message appears:

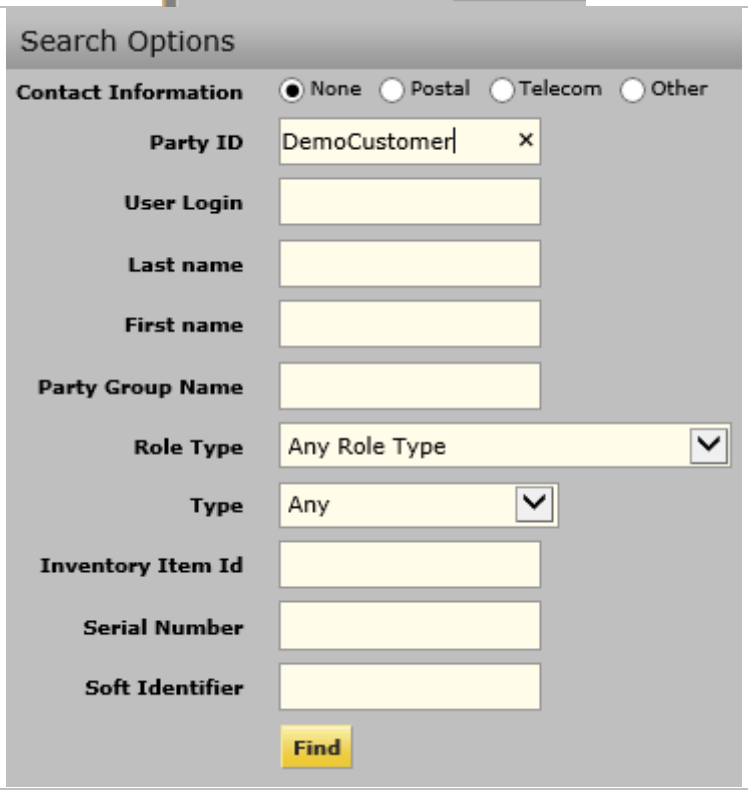


Party with more than one contact mechanism

Test Id	8
Purpose of the test	Ensure that a warning message is presented to the user when a given Party contains more than on contact mechanism (e.g. email, fax, telephone or website)
Test steps	Create or edit a given Party; Edit the Party so that is has at least two emails, two fax number, two telephones and two websites; Generate the SAF-T PT report.
Expected result	A warning message should be displayed saying that the Party has two contact mechanisms. The warning message should specify the correct Party and contact mechanism. The most recently created or updated contacts should be the ones on the report. The report should be valid according to the rules of the Report Validation Tool.

Go to the Party module and search for the party “DemoCustomer”.

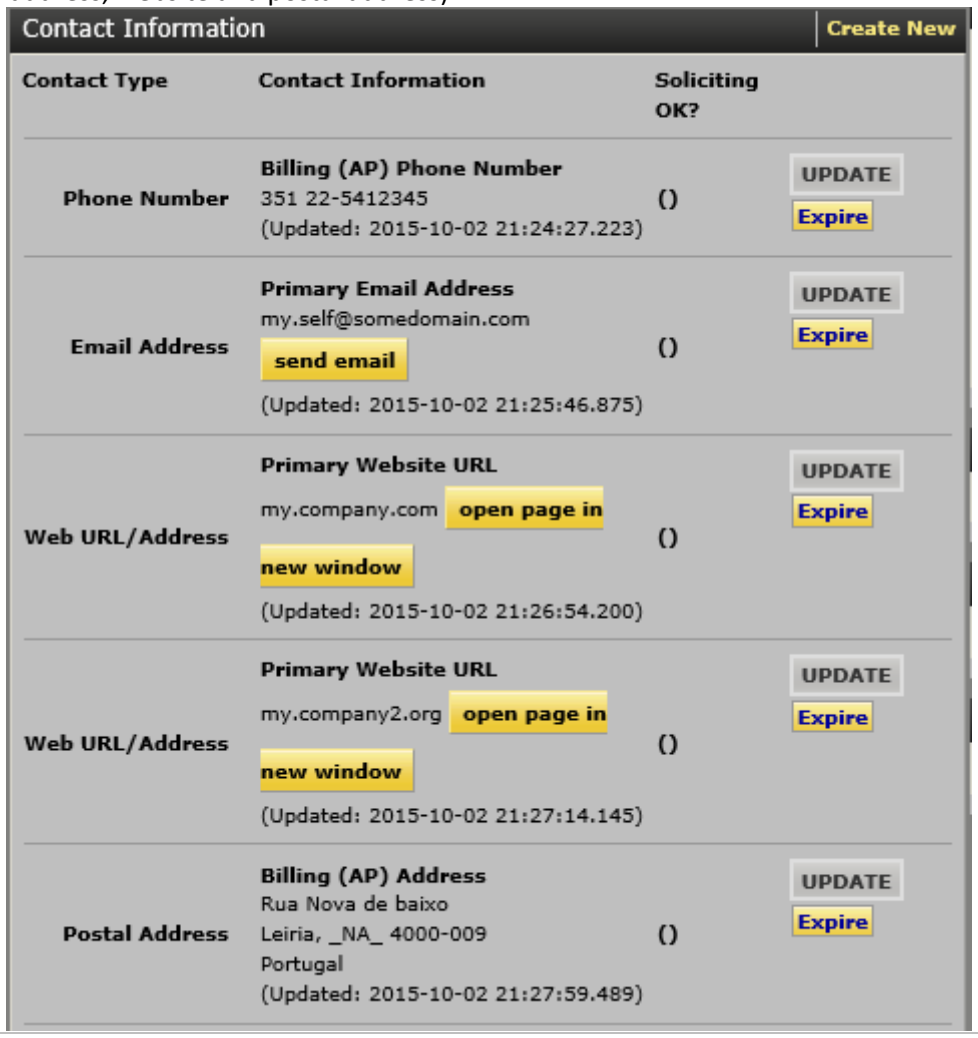


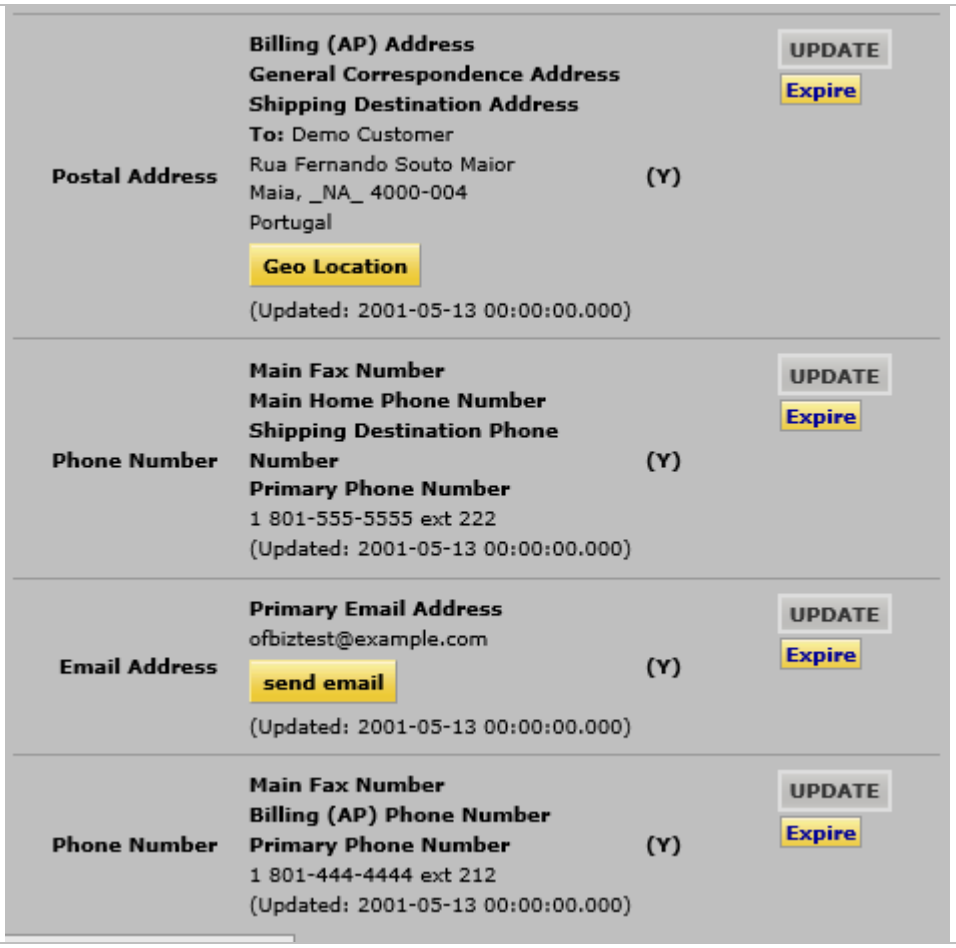


On the presented results table, open the “DemoCustomer” party.

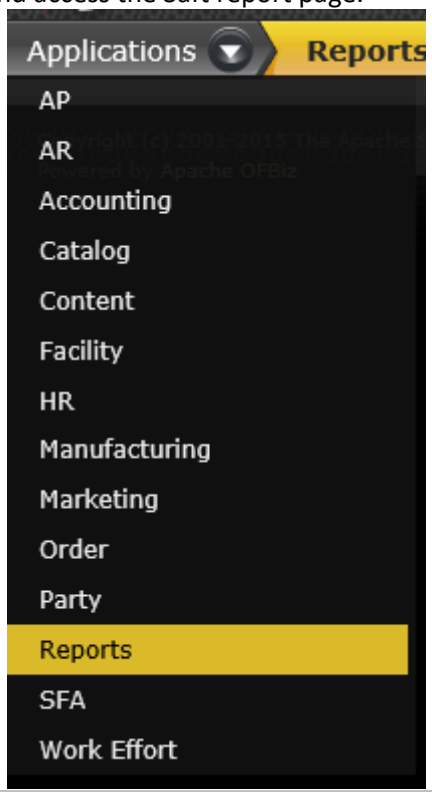
PARTY ID	USER LOGIN	NAME	RELATED COMPANY	TYPE	MAIN ROLE	CREATED DATE	LAST MODIFIED DATE	
DemoCustomer	DemoCustomer	Customer, Demo	Demo Customer Company	Person	Contact			DETAILS ORDERS QUOTES NEW ORDER NEW QUOTE
DemoCustomer2	(None)	(No name found)		???	Contact			DETAILS ORDERS QUOTES NEW ORDER NEW QUOTE
DemoCustomer1	(None)	(No name found)		???	Contact			DETAILS ORDERS QUOTES NEW ORDER NEW QUOTE
DemoCustomer3	(None)	(No name found)		???	Contact			DETAILS ORDERS QUOTES NEW ORDER NEW QUOTE

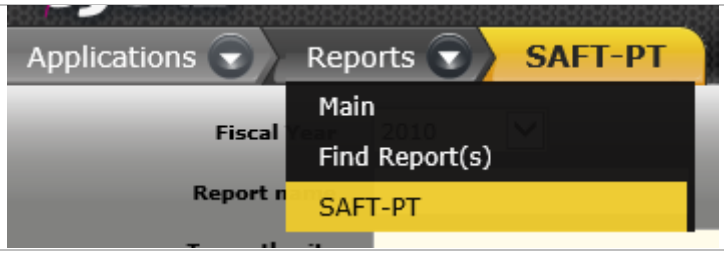
Make sure that the company has at least 2 contacts of each type (phone number and fax, email address, website and postal address):





Go to the Reports module and access the Saft report page.





Select the fiscal year for the current year and click “Submit”.

Check that the following warning messages appear for each type of duplicate contact mechanism:

Process identifier: 10041
Report type: Saft-PT
Report queued date: 2015-10-02 22:17:32.487
Process status: Finished

Download

Delete

MESSAGE SEVERITY	MESSAGE
Warning	More than one active postal address found for party with id 'Company' and country geo id 'PRT' within the specified fiscal year. The most recently updated one will be used.
Warning	More than one active postal address found for party with id 'DemoCustomer' and country geo id 'PRT' within the specified fiscal year. The most recently updated one will be used.
Warning	More than one contact mechanism was found for party 'DemoCustomer' with purpose type id 'FAX_NUMBER'.
Warning	More than one contact mechanism was found for party 'DemoCustomer' with purpose type id 'PRIMARY_PHONE'.
Warning	More than one contact mechanism was found for party 'DemoCustomer' with purpose type id 'PRIMARY_EMAIL'.
Warning	More than one contact mechanism was found for party 'DemoCustomer' with purpose type id 'PRIMARY_WEB_URL'.

Download the report and check that only one of those contacts appear in the xml file:

```
<Customer>
    <CustomerID>DemoCustomer</CustomerID>
    <AccountID>DemoCustomer</AccountID>
    <CustomerTaxID>705413726</CustomerTaxID>
    <CompanyName>Demo Customer</CompanyName>
    <BillingAddress>
        <AddressDetail>Rua Nova de baixo, Leiria, 4000-009</AddressDetail
        <City>Leiria</City>
        <PostalCode>4000-009</PostalCode>
        <Country>PT</Country>
    </BillingAddress>
    <Telephone>+1801444-4444</Telephone>
    <Fax>+1801444-4444</Fax>
    <Email>my.self@somedomain.com</Email>
    <Website>my.company2.org</Website>
    <SelfBillingIndicator>0</SelfBillingIndicator>
</Customer>
```

Check that the report is valid using the SAFT validation tool: