Apache OFBiz Developer Manual

The Apache OFBiz Project

Release 18.12

Table of Contents

1. Introduction

Welcome to the Apache OFBiz developer manual. This manual provides information to help with customizing and developing OFBiz. If you are new to OFBiz and interested in learning how to use it, you may want to start with the "Apache OFBiz User Manual".

OFBiz is a large system composed of multiple subsystems. This manual attempts to introduce the overall architecture and high level concepts, followed by a detailed description of each subsystem. In addition, the manual will cover topics necessary for developers including the development environment, APIs, deployment, security, and so on.

1.1. Main systems

OFBiz at its core is a collection of systems:

- A web server (Apache Tomcat)
- A web MVC framework for routing and handling requests.
- An entity engine to define, load and manipulate data.
- A service engine to define and control business logic.
- A widget system to draw and interact with a user interface.

On top of the above mentioned core systems, OFBiz provides:

- A data model shared across most businesses defining things like orders, invoices, general ledgers, customers and so on.
- A library of services that operate on the above mentioned data model such as "createBillingAccount" or "updateInvoice" and so on.
- A collection of applications that provide a user interface to allow users to interact with the system. These applications usually operate on the existing data model and service library. Examples include the "Accounting Manager" and "Order Manager".
- A collection of optional applications called "plugins" that extend basic functionality and is the main way to add custom logic to OFBiz.

1.2. Components

The basic unit in OFBiz is called "component". A component is at a minimum a folder with a file inside of it called "ofbiz-component.xml"

Every application in OFBiz is a component. For example, the order manager is a component, the accounting manager is also a component, and so on.

By convention, OFBiz components have the following main directory structure:

```
component-name-here/
 ├─── config/              - Properties and translation labels (i18n)
 ├─── data/                - XML data to load into the database
 ├─── entitydef/           - Defined database entities
 ├─── groovyScripts/       - A collection of scripts written in Groovy
 ├─── minilang/            - A collection of scripts written in minilang (deprecated)
 ├─── ofbiz-component.xml  - The OFBiz main component configuration file
 ├─── servicedef           - Defined services.
 ├─── src/
     ├─── docs/            - component documentation source
     └─── main/java/       - java source code
     └─── test/java/       - java unit-tests
 ├─── testdef              - Defined integration-tests
 ├─── webapp               - One or more Java webapps including the control servlet
 └─── widget               - Screens, forms, menus and other widgets
```

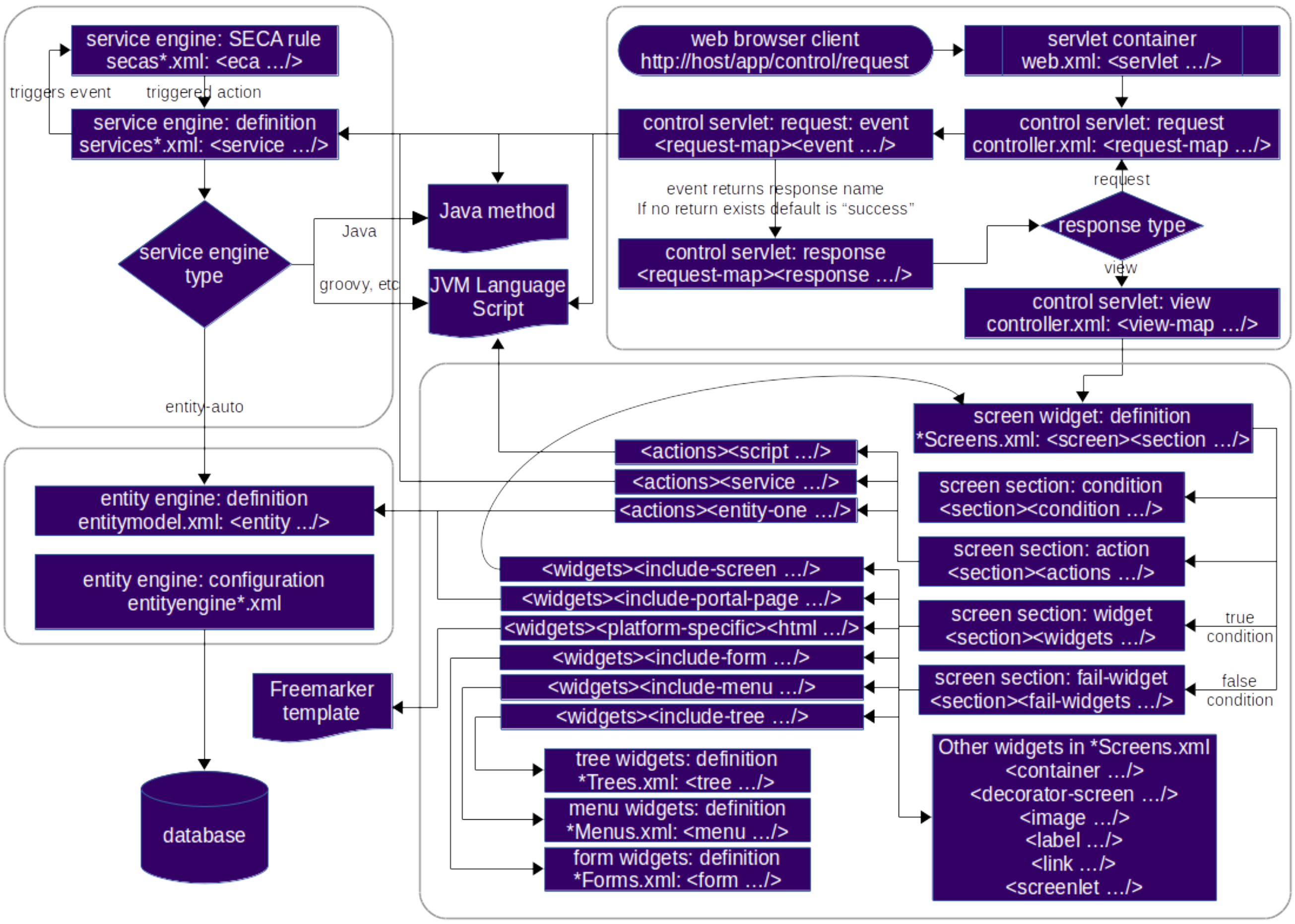
It is apparent from the above directory structure that each OFBiz component is in fact a full application as it contains entities, data, services, user interface, routing, tests, and business logic.

Both core OFBiz applications as well as plugins are nothing more than components. The only difference is that core applications reside in the "applications" folder whereas plugins reside in the "plugins" folder; also OFBiz does not ship with plugins by default.

1.3. Example workflow

Many basic concepts were explained so far. An example would help in putting all of these concepts together to understand the bigger picture. Let us take an example where a user opens a web browser and enters a certain URL and hits the enter key. What happens? It turns out answering this question is not quite simple because lots of things occur the moment the user hits "enter".

To try to explain what happens, take a look at the below diagram. Do not worry if it is not fully understandable, we will go through most of it in our example.


service engine: SECA rule
secas*.xml: <eca .../>
triggers event
triggered action
service engine: definition
services*.xml: <service .../>
service engine type
Java
groovy, etc
Java method
JVM Language Script
entity-auto
entity engine: definition
entitymodel.xml: <entity .../>
entity engine: configuration
entityengine*.xml
Freemarker template
database
web browser client
http://host/app/control/request
servlet container
web.xml: <servlet .../>
control servlet: request: event
<request-map><event .../>
control servlet: request
controller.xml: <request-map .../>
event returns response name
If no return exists default is "success"
request
response type
control servlet: response
<request-map><response .../>
view
control servlet: view
controller.xml: <view-map .../>
screen widget: definition
*Screens.xml: <screen><section .../>
<actions><script .../>
<actions><service .../>
<actions><entity-one .../>
screen section: condition
<section><condition .../>
screen section: action
<section><actions .../>
screen section: widget
<section><widgets .../>
true condition
screen section: fail-widget
<section><fail-widgets .../>
false condition
<widgets><include-screen .../>
<widgets><include-portal-page .../>
<widgets><platform-specific><html .../>
<widgets><include-form .../>
<widgets><include-menu .../>
<widgets><include-tree .../>
tree widgets: definition
*Trees.xml: <tree .../>
menu widgets: definition
*Menus.xml: <menu .../>
form widgets: definition
*Forms.xml: <form .../>
Other widgets in *Screens.xml
<container .../>
<decorator-screen .../>
<image .../>
<label .../>
<link .../>
<screenlet .../>

1.3.1. User enters URL

In the first step in our example, the user enters the following URL:

https://localhost:8443/accounting/control/findInvoices

If we break down this URL, we identify the following parts:

- localhost: Name of the server in which OFBiz is running
- 8443: Default https port for OFBiz
- accounting: web application name. A web application is something which is defined *inside* a component
- control: Tells OFBiz to transfer routing to the control servlet
- findInvoices: request name inside the control servlet

1.3.2. Control servlet takes over

The Java Servlet Container (tomcat) re-routes incoming requests through web.xml to a special OFBiz servlet called the control servlet. The control servlet for each OFBiz component is defined in controller.xml under the webapp folder.

The main configuration for routing happens in controller.xml. The purpose of this file is to map requests to responses.

Request Map

A request in the control servlet might contain the following information:

- Define communication protocol (http or https) as well as whether authentication is required.
- Fire up an event which could be either a piece of code (like a script) or a service.
- Define a response to the request. A response could either be another request or a view map.

So in this example, the findInvoices request is mapped to a findInvoices view.

View Map

A view map maps a view name to a certain view-type and a certain location.

View types can be one of:

- screen: A screen widget which translates to normal HTML.
- screenfop: A PDF screen designed with Apache FOP based constructs.
- screencsv: A comma separated value output report.
- screenxml: An XML document.
- simple-content; A special MIME content type (like binary files).
- ftl: An HTML document generated directly from a FreeMarker template.
- screenxls: An Excel spreadsheet.

In the findInvoices example, the view-map type is a normal screen which is mapped to the screen: component://accounting/widget/InvoiceScreens.xml#FindInvoices

1.3.3. Widget rendered

Once the screen location is identified and retrieved from the previous step, the OFBiz widget system starts to translate the XML definition of the screen to actual HTML output.

A screen is a collection of many different things and can include:

- Other screens
- Decorator screens
- Conditional logic for hiding / showing parts of the screen
- data preparation directives in the <action> tag
- Forms
- Menus
- Trees
- Platform specific code (like FreeMarker for HTML output)
- Others (portals, images labels etc …)

Continuing the example, the FindInvoices screen contains many details including two forms. One form is for entering invoice search fields and the other form displays search results.

2. Web Framework

3. Web Applications

The OFBiz webapp is one of the core framework components. It is tightly integrated with other framework components.

3.1. Cross-domains Single Sign On (SSO)

In some cases you need to split the OFBiz applications on different servers, and possibly in production on different domains. This can happen for different reasons, most often for performance reason.

As it’s annoying to give each time a credential when changing from an OFBiz application to another on the same server, the same applies when changing from an OFBiz application to another on another domain.

To prevent that on the same server, the ExternalLoginKey mechanism is used. The cross-domains SSO feature allows to navigate from a domain to another with automated SSO.

It based on 3 technologies:

JWT
: JWT Official site - Wikipedia for JWT

CORS
: CORS (Mozilla doc) - Wikipedia for CORS

Ajax
: Ajax, now well known I guess, in OFBiz we use jQuery for that.

The mechanism is simple.

On the source side:

1. When an user log in in an application (webApp) a webappName.securedLoginId cookie is created. This cookie will be used by the mechanism to know the current logged in user. *Note that all webappName.securedLoginId cookies are deleted when the user session is closed or time out. Hence (apart also using an intrinsically secured cookie) the mechanim is secured, even on shared machines. Of course if people are sharing a machine during their sessions, things could get complicated. This unlikely later case is not taken in account.*
2. The user is given a JavaScript link which passes the URL to reach and the calling webapp name to the sendJWT() Ajax function.
3. The sendJWT() Ajax function calls the loadJWT() Ajax function which in turn calls the CommonEvents::loadJWT method through the common controller.
4. The CommonEvents::loadJWT method uses the calling webapp name to retrieve the userLoginId from the secured webappName.securedLoginId cookie, creates a JWT containing the userLoginId, and returns it to the loadJWT() Ajax function.
5. Then the sendJWT() Ajax function sends an Authorization header containing the JWT to the URL

to reach. At this stage, if all things are correct, the flow leaves the source side.

On the server side:

1. A CORS policy is needed. *Without it, the Authorization token containing the JWT will be rejected. It's a simple policy but you need to strictly define the authorized domains. Never use the lazy "*" for domains (ie all domains), else the preflight request will not work.* Here is an example for Apache HTTPD (domain value is "https://localhost:8443" for official OFBiz demo):

```
Header set Access-Control-Allow-Origin domain
Header set Access-Control-Allow-Headers "Authorization"
Header set Access-Control-Allow-Credentials "true"
```

1. The checkJWTLogin preprocessor, similar to the checkExternalLoginKey, intercepts the JWT, checks it and if all is OK signs the user on. That's it !

In the example component, the FormWidgetExamples screen contains 2 new fields in the LinksExampleForm which demonstrate the use from a local instance to the trunk demo instance.

If you are interested in more details you may refer to https://issues.apache.org/jira/browse/OFBIZ-10307

3.2. Control Servlet

3.2.1. Requests

3.2.2. Views

4. Entity Engine

4.1. Entities

4.1.1. Standard Entities

4.1.2. View Entities

4.1.3. Extended Entities

4.1.4. Dynamic View Entities

4.2. XML Data

4.3. Entity engine configuration

4.4. Supported databases

4.5. Data Model Changes

The Apache OFBiz® Project Release 18.12

Apache OFBiz follows **The Universal Data Model** by **Len Silverston**, with a grain of salt.

The following file contains information about the data model changes in the Apache OFBiz. The detailed description of migration scripts specified here can be found at Revisions Requiring Data Migration - upgrade ofbiz page.

4.5.1. Changes with OFBiz Trunk (Upcoming Branch)

Entity Changes

Added 1 new entity

1. ProdPromoCodeContactMech

Removed/Deprecate 1 entity

1. ProductPromoCodeEmail

Field Changes

No changes

Migration Scripts

1. Migration service migrateProductPromoCodeEmail is implemented to migrate the ProductPromoCodeEmail entity to ProductPromoCodeContactMech.
   (More detail at OFBIZ-5426)

4.5.2. Changes with OFBiz 17

Field types `id-ne`, `id-long-ne` & `id-vlong-ne` has been removed. Use `id`, `id-long` and `id-vlong` instead (detailed description at OFBIZ-9351).

Entity Changes

No changes

Field Changes

Entity	Field	Action	IsPK	Revision
MarketingCampaignPrice	fromDate	Added	Yes	R1805961
MarketingCampaignPrice	thruDate	Added	No	R1805961
MarketingCampaignPromo	fromDate	Added	Yes	R1805961
MarketingCampaignPromo	thruDate	Added	No	R1805961
MarketingCampaignRole	fromDate	Added	Yes	R1805961
MarketingCampaignRole	thruDate	Added	No	R1805961
Product	manufacturerPartyId	Removed	No	R1804408
SecurityGroupPermission	fromDate	Added	Yes	R1812383
SecurityGroupPermission	thruDate	Added	No	R1812383

Migration Scripts

1. Updated sql-type for date-time and time field in fieldtypemysql.xml file at commit R1793300
   Update msyql sql-type for datetime field-type to support Fractional Seconds in Time Values Please upgrade mysql to at least 5.6.4 or higher.
   After upgrade run `generateMySqlFileWithAlterTableForTimestamps` service, groupName is required field for this service.
   It will generate sql file with alter query statement for date-time and time field at location `$\{ofbiz.home}/runtime/tempfiles/.sql`
   You can use execute sql statement from any of the mysql batch command.

4.5.3. Changes between OFBiz 9 to OFBiz 16

Entity Changes

Added 77 new entities

1. JobRequisition
2. ProductAverageCostType
3. WorkEffortSurveyAppl
4. WorkEffortIcalData
5. WebSiteContactList
6. WebAnalyticsType
7. WebAnalyticsConfig
8. UserLoginSecurityQuestion
9. UomGroup
10. TrainingRequest
11. ThirdPartyLogin
12. TestFieldType
13. TestingSubtype
14. TestingStatus
15. TestingRemoveAll
16. TestingItem
17. TestingCrypto
18. SystemProperty
19. ShipmentGatewayUsps
20. ShipmentGatewayUps
21. ShipmentGatewayFedex
22. ShipmentGatewayDhl
23. ShipmentGatewayConfig
24. ShipmentGatewayConfigType
25. ReturnContactMech
26. QuoteNote
27. ProductPromoContent
28. ProductPromoContentType
29. ProductGroupOrder
30. ProductCostComponentCalc
31. CostComponentCalc

32. PayPalPaymentMethod
33. PaymentGroupType
34. PaymentGroup
35. PaymentGroupMember
36. PaymentGatewayConfig
37. PaymentGatewayConfigType
38. PaymentGatewayWorldPay
39. PaymentGatewaySecurePay
40. PaymentGatewaySagePay
41. PaymentGatewayOrbital
42. PaymentGatewayEway
43. PaymentGatewayCyberSource
44. PaymentGatewayAuthorizeNet
45. PaymentGatewayIDEAL
46. PaymentContentType
47. PaymentContent
48. OAuth2LinkedIn
49. OAuth2GitHub
50. JobManagerLock
51. JobInterviewType
52. JobInterview
53. JavaResource
54. InvoiceNote
55. InvoiceItemAssocType
56. InvoiceItemAssoc
57. InvoiceContentType
58. InvoiceContent
59. GlAccountCategoryType
60. GlAccountCategoryMember
61. GlAccountCategory
62. GitHubUser
63. FixedAssetTypeGlAccount
64. FacilityContent
65. ExcelImportHistory
66. EmplLeaveReasonType

67. EbayShippingMethod
68. EbayConfig
69. CountryAddressFormat
70. ContentSearchResult
71. ContentSearchConstraint
72. ContentKeyword
73. CheckAccount
74. AgreementFacilityAppl
75. AgreementContentType
76. AgreementContent

Removed 8 entities

1. DepreciationMethod
2. FixedAssetMaintMeter
3. OagisMessageErrorInfo
4. OagisMessageInfo
5. SalesOpportunityTrackingCode
6. SimpleSalesTaxLookup
7. TestBlob
8. WorkEffortAssignmentRate

Field Changes

Entity	Field	Action	IsPK	Revision
AcctgTransAttribute	attrDescription	Added	No	NA
AcctgTransEntry	inventoryItemId	Added	No	NA
AcctgTransTypeAttr	description	Added	No	NA
BenefitType	parentTypeId	Added	No	NA
BenefitType	hasTable	Added	No	NA
BudgetAttribute	attrDescription	Added	No	NA
BudgetItemAttribute	attrDescription	Added	No	NA
BudgetItemTypeAttr	description	Added	No	NA
BudgetStatus	changeByUserLoginId	Added	No	NA
BudgetTypeAttr	description	Added	No	NA

Entity	Field	Action	IsPK	Revision
CommunicationEventRole	statusId	Added	No	NA
CommunicationEventType	contactMechTypeId	Added	No	NA
ContactListCommStatus	partyId	Added	No	NA
ContactListCommStatus	messageId	Added	No	NA
ContactListCommStatus	changeByUserLoginId	Added	No	NA
ContactMechAttribute	attrDescription	Added	No	NA
ContactMechTypeAttr	description	Added	No	NA
DeductionType	parentTypeId	Added	No	NA
DeductionType	hasTable	Added	No	NA
DocumentAttribute	attrDescription	Added	No	NA
DocumentTypeAttr	description	Added	No	NA
EmploymentApp	approverPartyId	Added	No	NA
EmploymentApp	jobRequisitionId	Added	No	NA
EmploymentAppSourceType	parentTypeId	Added	No	NA
EmploymentAppSourceType	hasTable	Added	No	NA
EmplPositionClassType	parentTypeId	Added	No	NA
EmplPositionClassType	hasTable	Added	No	NA
EmplPositionType	parentTypeId	Added	No	NA
EmplPositionType	hasTable	Added	No	NA
EmplPositionType	partyId	Removed	No	NA
EmplPositionType	roleTypeId	Removed	No	NA
FinAccountAttribute	attrDescription	Added	No	NA
FinAccountTransAttribute	attrDescription	Added	No	NA
FinAccountTrans	glReconciliationId	Added	No	NA
FinAccountTrans	statusId	Added	No	NA

Entity	Field	Action	IsPK	Revision
FinAccountTransTypeAttr	description	Added	No	NA
FinAccountTypeAttr	description	Added	No	NA
FinAccountStatus	changeByUserLoginId	Added	No	NA
FixedAsset	acquireOrderId	Added	No	NA
FixedAsset	acquireOrderItemSeqId	Added	No	NA
FixedAssetAttribute	attrDescription	Added	No	NA
FixedAssetTypeAttr	description	Added	No	NA
GlAccount	externalId	Added	No	NA
GlAccount	openingBalance	Added	No	NA
GlReconciliation	createdDate	Added	No	NA
GlReconciliation	lastModifiedDate	Added	No	NA
GlReconciliation	statusId	Added	No	NA
GlReconciliation	openingBalance	Added	No	NA
InventoryItemAttribute	attrDescription	Added	No	NA
InventoryItemStatus	changeByUserLoginId	Added	No	NA
InventoryItemTypeAttr	description	Added	No	NA
InvoiceAttribute	attrDescription	Added	No	NA
InvoiceItemAttribute	attrDescription	Added	No	NA
InvoiceItemTypeAttr	description	Added	No	NA
InvoiceStatus	changeByUserLoginId	Added	No	NA
InvoiceTypeAttr	description	Added	No	NA
InvoiceTermAttribute	attrDescription	Added	No	NA
JobSandbox	currentRetryCount	Added	No	NA
JobSandbox	tempExprId	Added	No	NA
JobSandbox	currentRecurrenceCount	Added	No	NA

Entity	Field	Action	IsPK	Revision
JobSandbox	maxRecurrenceCount	Added	No	NA
JobSandbox	jobResult	Added	No	NA
OrderAdjustment	amountAlreadyIncluded	Added	No	NA
OrderAdjustment	isManual	Added	No	NA
OrderAdjustment	oldPercentage	Added	No	NA
OrderAdjustment	oldAmountPerQuantity	Added	No	NA
OrderAdjustment	lastModifiedDate	Added	No	NA
OrderAdjustment	lastModifiedByUserLogin	Added	No	NA
OrderAdjustmentAttribute	attrDescription	Added	No	NA
OrderAdjustmentTypeAttr	description	Added	No	NA
OrderAttribute	attrDescription	Added	No	NA
OrderItem	supplierProductId	Added	No	NA
OrderItem	cancelBackOrderDate	Added	No	NA
OrderItem	changeByUserLoginId	Added	No	NA
OrderItemAttribute	attrDescription	Added	No	NA
OrderItemShipGroup	facilityId	Added	No	NA
OrderItemShipGroup	estimatedShipDate	Added	No	NA
OrderItemShipGroup	estimatedDeliveryDate	Added	No	NA
OrderItemShipGrpInvRes	priority	Added	No	NA
OrderItemShipGrpInvRes	oldPickStartDate	Added	No	NA
OrderItemTypeAttr	description	Added	No	NA
OrderTermAttribute	attrDescription	Added	No	NA
OrderPaymentPreference	track2	Added	No	NA

Entity	Field	Action	IsPK	Revision
OrderPaymentPreference	swipedFlag	Added	No	NA
OrderPaymentPreference	lastModifiedDate	Added	No	NA
OrderPaymentPreference	lastModifiedByUserLogin	Added	No	NA
OrderShipment	shipGroupSeqId	Added	No	NA
OrderTypeAttr	description	Added	No	NA
PartyAcctgPreference	orderSequenceEnumId	Removed	No	NA
PartyAcctgPreference	quoteSequenceEnumId	Removed	No	NA
PartyAcctgPreference	invoiceSequenceEnumId	Removed	No	NA
PartyAcctgPreference	oldOrderSequenceEnumId	Added	No	NA
PartyAcctgPreference	oldQuoteSequenceEnumId	Added	No	NA
PartyAcctgPreference	oldInvoiceSequenceEnumId	Added	No	NA
PartyAcctgPreference	orderSeqCustMethId	Added	No	NA
PartyQual	infoString	Removed	No	NA
PartyQual	institutionInternalId	Removed	No	NA
PartyQual	institutionPartyId	Removed	No	NA
PartyQual	partyQualId	Removed	No	NA
PartyRate	percentageUsed	Added	No	NA
PartyRate	rate	Removed	No	NA
PartyResume	contentId	Added	No	NA
PaymentAttribute	attrDescription	Added	No	NA
PaymentGatewayResponse	gatewayCvResult	Added	No	NA
PaymentMethod	finAccountId	Added	No	NA
PaymentTypeAttr	description	Added	No	NA
PerfRatingType	parentTypeId	Added	No	NA
PerfRatingType	hasTable	Added	No	NA
PerfReview	payHistoryRoleTypeIdTo	Removed	No	NA

Entity	Field	Action	IsPK	Revision
PerfReview	payHistoryRoleTypeIdFrom	Removed	No	NA
PerfReview	payHistoryPartyIdTo	Removed	No	NA
PerfReview	payHistoryPartyIdFrom	Removed	No	NA
PerfReview	payHistoryFromDate	Removed	No	NA
PerfReviewItemType	parentTypeId	Added	No	NA
PerfReviewItemType	hasTable	Added	No	NA
PersonTraining	trainingRequestId	Added	No	NA
PersonTraining	workEffortId	Added	No	NA
PersonTraining	approverId	Added	No	NA
PersonTraining	approvalStatus	Added	No	NA
PersonTraining	reason	Added	No	NA
PostalAddress	houseNumber	Added	No	NA
PostalAddress	houseNumberExt	Added	No	NA
PostalAddress	cityGeoId	Added	No	NA
PostalAddress	municipalityGeoId	Added	No	NA
PostalAddress	geoPointId	Added	No	NA
PosTerminal	terminalName	Added	No	NA
PosTerminalInternTx	reasonEnumId	Added	No	NA
Product	releaseDate	Added	No	NA
Product	originalImageUrl	Added	No	NA
Product	inventoryItemTypeId	Added	No	NA
Product	shippingWeight	Added	No	NA
Product	productWeight	Added	No	NA
Product	diameterUomId	Added	No	NA
Product	productDiameter	Added	No	NA
Product	virtualVariantMethodEnum	Added	No	NA
Product	defaultShipmentBoxTypeId	Added	No	NA
Product	lotIdFilledIn	Added	No	NA

Entity	Field	Action	IsPK	Revision
Product	orderDecimalQuantity	Added	No	NA
Product	weight	Removed	No	NA
Product	taxCategory	Removed	No	NA
Product	taxVatCode	Removed	No	NA
Product	taxDutyCode	Removed	No	NA
ProductAttribute	attrDescription	Added	No	NA
ProductAverageCost	productAverageCostTypeId	Added	No	NA
ProductAverageCost	facilityId	Added	No	NA
ProductContent	sequenceNum	Added	No	NA
ProductKeyword	keywordTypeId	Added	No	NA
ProductKeyword	statusId	Added	No	NA
ProductRole	sequenceNum	Added	No	NA
ProductStore	balanceResOnOrderCreation	Added	No	NA
ProductStore	defaultTimeZoneString	Added	No	NA
ProductStore	oldStyleSheet	Added	No	NA
ProductStore	oldHeaderLogo	Added	No	NA
ProductStore	oldHeaderRightBackground	Added	No	NA
ProductStore	oldHeaderMiddleBackground	Added	No	NA
ProductStore	styleSheet	Removed	No	NA
ProductStore	headerLogo	Removed	No	NA
ProductStore	headerRightBackground	Removed	No	NA
ProductStore	headerMiddleBackground	Removed	No	NA
ProductStorePaymentSetting	paymentCustomMethodId	Added	No	NA
ProductStorePaymentSetting	paymentGatewayConfigId	Added	No	NA
ProductStoreShipmentMeth	shipmentCustomMethodId	Added	No	NA
ProductStoreShipmentMeth	shipmentGatewayConfigId	Added	No	NA

Entity	Field	Action	IsPK	Revision
ProductStoreShipmentMeth	allowancePercent	Added	No	NA
ProductStoreShipmentMeth	minimumPrice	Added	No	NA
ProductTypeAttribute	attrDescription	Added	No	NA
QuoteAdjustment	lastModifiedDate	Added	No	NA
QuoteAdjustment	lastModifiedByUserLogin	Added	No	NA
QuoteAttribute	attrDescription	Added	No	NA
QuoteItem	leadTimeDays	Added	No	NA
QuoteRole	fromDate	Added	Yes	NA
QuoteRole	thruDate	Added	No	NA
QuoteTerm	termDays	Added	No	NA
QuoteTerm	textValue	Added	No	NA
QuoteTerm	description	Added	No	NA
QuoteTermAttribute	attrDescription	Added	No	NA
QuoteTypeAttr	description	Added	No	NA
RequirementAttribute	changeByUserLoginId	Added	No	NA
RequirementStatus	changeByUserLoginId	Added	No	NA
ResponsibilityType	parentTypeId	Added	No	NA
ResponsibilityType	hasTable	Added	No	NA
ReturnAdjustment	createdByUserLoginId	Added	No	NA
ReturnAdjustment	lastModifiedDate	Added	No	NA
ReturnAdjustment	lastModifiedByUserLogin	Added	No	NA
ReturnHeader	supplierRmaId	Added	No	NA
ReturnItemResponse	finAccountTransId	Added	No	NA
ReturnStatus	changeByUserLoginId	Added	No	NA
SalaryStep	fromDate	Added	Yes	NA
SalaryStep	thruDate	Added	No	NA

Entity	Field	Action	IsPK	Revision
SalaryStep	createdByUserLoginId	Added	No	NA
SalaryStep	lastModifiedByUserLogin	Added	No	NA
SalesOpportunity	nextStepDate	Added	No	NA
ServiceSemaphore	lockedByInstanceId	Added	No	NA
ShoppingListItem	modifiedPrice	Added	No	NA
SkillType	parentTypeId	Added	No	NA
SkillType	hasTable	Added	No	NA
SupplierProduct	shippingPrice	Added	No	NA
SupplierProduct	supplierCommissionPerc	Removed	No	NA
TaxAuthorityRateProduct	isTaxInShippingPrice	Added	No	NA
TerminationType	parentTypeId	Added	No	NA
TerminationType	hasTable	Added	No	NA
TestingNodeMember	extendFromDate	Added	No	NA
TestingNodeMember	extendThruDate	Added	No	NA
TimeEntry	planHour	Added	No	NA
Timesheet	approvedByUserLoginId	Added	No	NA
TrainingClassType	parentTypeId	Added	No	NA
TrainingClassType	hasTable	Added	No	NA
UnemploymentClaim	thruDate	Added	No	NA
UserLogin	externalAuthId	Added	No	NA
UserLogin	userLdapDn	Added	No	NA
UserLogin	disabledBy	Added	No	NA
ValueLinkKey	createdByUserLogin	Added	No	NA
WebSite	visualThemeSetId	Added	No	NA
WebSite	hostedPathAlias	Added	No	NA
WebSite	isDefault	Added	No	NA
WebSite	displayMaintenancePage	Added	No	NA
WebSitePathAlias	fromDate	Added	Yes	R1738588

Entity	Field	Action	IsPK	Revision
WebSitePathAlias	thruDate	Added	No	R1738588
WorkEffort	tempExprId	Added	No	NA
WorkEffort	sequenceNum	Added	No	NA
WorkEffortAttribute	attrDescription	Added	No	NA
WorkEffortAssocAttribute	attrDescription	Added	No	NA
WorkEffortAssocTypeAttr	description	Added	No	NA
WorkEffortContactMech	fromDate	Added	Yes	NA
WorkEffortContactMech	thruDate	Added	No	NA
WorkEffortFixedAssetAssign	availabilityStatusId	Added	No	NA
WorkEffortPartyAssignment	assignedByUserLoginId	Added	No	NA
WorkEffortPurposeType	parentTypeId	Added	No	NA
WorkEffortStatus	reason	Added	No	NA
WorkEffortTypeAttr	description	Added	No	NA
WorkOrderItemFulfillment	shipGroupSeqId	Added	No	NA

5. Service Engine

5.1. Declaration and Implementation

5.2. Supported languages

5.3. Transaction management

5.4. Web services

6. Widget System

6.1. Screen Widget

6.1.1. Decoration Pattern

6.2. Form Widget

6.3. Menu Widget

6.4. Tree Widget

6.5. Portal Widget

6.6. Platform Specific Code

7. Core APIs

8. Development environment

8.1. Setup your environment

8.1.1. Java SE

8.1.2. IDE

Eclipse

Intellij Idea

8.1.3. Database

8.2. Web tools

9. Testing

9.1. Unit Tests

9.2. Integration Tests

10. Deployment

11. Security

11.1. CSRF defense

11.1.1. How is done the CSRF defense in Apache OFBiz and how to adapt it if needed

The Apache OFBiz Project Release 18.12

The same-Site attribute

> The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
>
> — According to OWASP ZAP

By default OOTB the SameSiteFilter property sets the same-site attribute value to 'strict. SameSiteFilter allows to change to 'lax' if needed

Properties

The *security.properties* file contains related properties:

```
# -- By default the SameSite value in SameSiteFilter is 'strict'.
# -- This property allows to change to 'lax' if needed.
SameSiteCookieAttribute=
```

11.2. Passwords and JWT (JSON Web Tokens) usage

11.2.1. How are set and used passwords and JWT in Apache OFBiz

The Apache OFBiz Project Release 18.12

Passwords

Demo and seed passwords are stored in files loaded through security ofbiz-component.xml. To know more about that be sure to read:

- The technical production setup guide notably "Initial Data Loading" and "Security Settings" sections
- How to secure your deployment

These configuration steps are not to be neglected for the security of a **production environment**

JWT usage

As says Wikipedia:

> JSON Web Token (JWT) is an Internet standard for creating JSON-based access tokens that assert some number of claims.

We currently use JWT in 2 places:

1. To let users safely recreate passwords (in backend and frontend)
2. To allow SSO (Single Sign-on) jumpings from an OFBiz instance to another on another domain, by also using CORS (Cross-origin resource sharing) on the target server

How to secure JWT

When you use JWT, in order to sign your tokens, you have the choice of using a sole so called secret key or a pair of public/private keys: https://jwt.io/introduction/.

You might prefer to use pair of public/private keys, for now by default OFBiz uses a simple secret key. Remains the way how to store this secret key. This is an interesting introduction about this question.

1. The first idea which comes to mind is to use a property in the security.properties file. It's safe as long as your file system is not compromised.
2. You may also pick a SystemProperty entity (overrides the file property). It's safe as long as your DB is not compromised.
3. We recommend to not use an environment variable as those can be considered weak:
   - http://movingfast.io/articles/environment-variables-considered-harmful
   - https://security.stackexchange.com/questions/49725/is-it-really-secure-to-store-api-keys-in-environment-variables
4. You may want to tie the encryption key to the logged in user. This is used by the password recreation feature. The JWT secret key is salted with a combination of the current logged in user and her/his password. This is a simple and effective safe way.
5. Use a JTI (JWT ID). A JTI prevents a JWT from being replayed. This auth0 blog article get deeper in that. The same is kinda achieved with the password recreation feature. When the user log in after the new password creation, the password has already been changed. So the link (in the sent email) containing the JWT for the creation of the new password can't be reused.
6. Tie the encryption key to the hardware. You can refer to this Wikipedia page for more information.
7. If you want to get deeper in this get to this OWASP documentation

Note: if you want to use a pair of public/private keys you might want to consider leveraging the Java Key Store that is also used by the "catalina" component to store certificates. Then don't miss to read:

- https://cryptosense.com/blog/mighty-aphrodite-dark-secrets-of-the-java-keystore/
- https://neilmadden.blog/2017/11/17/java-keystores-the-gory-details/

Also remember that like everything a JWT can be attacked and, though not used or tried in OFBiz yet, a good way is to mitigate an attack by using a KeyProvider. I have created OFBIZ-11187 for that.

Properties

The *security.properties* file contains five related properties:

```
# -- If false, then no externalLoginKey parameters will be added to cross-webapp urls
security.login.externalLoginKey.enabled=true
```

```
# -- Security key used to encrypt and decrypt the autogenerated password in forgot
password functionality.
login.secret_key_string=login.secret_key_string
```

```
# -- Time To Live of the token send to the external server in seconds, 10 seconds
seems plenty enough OOTB. Custom projects might want set a lower value.
security.jwt.token.expireTime=1800
```

```
# -- Enables the internal Single Sign On feature which allows a token based login
between OFBiz instances
# -- To make this work you also have to configure a secret key with security.token.key
security.internal.sso.enabled=false
```

```
# -- The secret key for the JWT token signature. Configuration in the SystemProperty
entity is recommended for security reasons.
security.token.key=security.token.key
```

Last but not least

Be sure to read Keeping OFBiz secure

11.3. Impersonation

11.3.1. What is Impersonation in Apache OFBiz

The Apache OFBiz Project Release 18.12

Introduction to User impersonation

User Impersonation is a feature that offer a way to select a user login and impersonate it, i.e. see what the user could see navigating through the application in his name.

How do this work ?

An authorized user *(see security and controls section for configuration)*, can select a user that will be impersonated.

The impersonation start, if everything is well configured, in current application (partymgr for the demo). Everything appears like if we were logged in with the userLoginId and the valid password (though we know nothing about it)

The only thing showing that we currently are impersonating a user is the little bottom-right image :

This icon indicates, when clicking on it, the user impersonated, and offer a way to depersonate.

The impersonate period is stored for audit purpose, and if the impersonator forgot to depersonate, the period is terminated *one hour* after impersonation start.

Security

This feature can draw some concerns about security aspect. This paragraph will introduce every controls and properties that have been implemented around the impersonation feature.

These configuration steps are not to be neglected for a **production environment** since this feature offer a way to act in place of another user.

Properties

The *security.properties* file introduce two properties that control impersonation feature :

```
security.disable.impersonation = true
```

This property, set by default to **true**, controls the activation of impersonation feature. If no configuration is done any user trying to use impersonation will face an error message, indicating that the feature is disabled.

To enable impersonation this property need to be set to **false**

```
security.login.authorised.during.impersonate = false
```

This property controls the way impersonation occurred to the impersonated user :

In default configuration, the impersonated user see nothing and can use the application without knowing that he is currently impersonated. Several authorized user can impersonate a same login without any issue.

This configuration is intended for testing/QA environment allowing any authorized user to impersonate a login to validate its configuration, test the application etc.

Set to **true**, this configuration improve the control of the data generated by the impersonated user. Indeed, Only one authorized user can impersonate a login at the same time, and during the impersonation process, the impersonated user is unable to act within the application.

Since the impersonation period is stored in database, the actions done by the authorized user can be identified if there is the need to do so.

This configuration is intended for production environment

Controls

The permission

First, to be able to use impersonation, a user need to possess *IMPERSONATE_ADMIN* permissions. Demo data offer *IMPERSONATION* security group for this purpose.
In demo data, *FULLADMIN* security group also possess the permission.

Permission based user restriction

An authorized user cannot impersonate any user. There are two main controls that will restrict the impersonation feature.

Cannot impersonate Admin user

It is impossible to impersonate a user that is granted any of the admin permission :

```
"IMPERSONATE_ADMIN"
"ARTIFACT_INFO_VIEW"
"SERVICE_MAINT"
"ENTITY_MAINT"
"UTIL_CACHE_VIEW"
"UTIL_DEBUG_VIEW"
```

Cannot impersonate more privileged user

It is impossible to impersonate a user that has more permission than your user. Even if the missing persmission is a minor one.

12. Appendices

13. From Mini Language to Groovy

This is a small guide for everybody involved in converting the Mini Language into Groovy.

Why is this important?
This tutorial is directly linked to the efforts of converting all scripts in Mini Language to newer Groovy Scripts. All of this is done, because Groovy is much more readable and easier to review, more up to date and many other reasons, which can be found here: Proposal for deprecating Mini Language

To contribute, or just be up to date with the current process, you can look at the existing JIRA issue OFBIZ-9350 - Deprecate Mini Lang

13.1. Groovy DSL (dynamic scripting library)

13.1.1. How to get Groovy support in your IDE

The following paragraph is for Eclipse users.

It is possible to get Groovy support in Eclipse by converting the loaded project to a Groovy Project. The project itself will work as before.

To do this just follow these few steps:

1. Right-click on the project that has to be converted
2. Click on "Configure"
3. Click on "Convert to Groovy Project"

Eclipse will automatically load the file OfbizDslDescriptorForEclipse.dsld , in which the known fields and methods used in Groovy Scripts are defined.

13.1.2. Known Fields

`property name: 'parameters'`
`type : 'java.util.Map'`
These are the parameters given to the Groovy Script, when it is called as a service. It is equivalent to `Map<String, Object>` context in the Java-Service-Definition.

`property name: 'context'`
`type: 'java.util.Map'`
More parameters, which are, for example, given through a screen or another Groovy Script. This is important when the script is called through an action segment of a screen.

`property name: 'delegator'`
`type: 'org.apache.ofbiz.entity.Delegator'`
Normal instance of the Delegator, which is used for special database access.

`property name: 'dispatcher'`
`type: 'org.apache.ofbiz.service.LocalDispatcher'`
Normal instance of the LocalDispatcher, which is used to call services and other service-like operations.

`property name: 'security'`
`type: 'org.apache.ofbiz.security.Security'`
Normal instance of the Security-Interface with which permission checks are done.

13.2. Known Methods

`method name: 'runService'`
`type: 'java.util.Map'`
`params: [serviceName: 'String', inputMap: 'java.util.Map']`
Helping method to call services instead of dispatcher.runSync(serviceName, inputMap). Also possible: run service: serviceName, with: inputMap

`method name: 'makeValue'`
`type: 'java.util.Map'`
`params: [entityName: 'String']`
Helping method to make a GenericValue instead of delegator.makeValue(entityName). Creates an empty GenericValue of the specific entity.

`method name: 'findOne'`
`type: 'java.util.Map'`
`params: [entityName: 'String', inputMap: 'java.util.Map']`
Helping method to find one GenericValue in the database. Used instead of delegator.findOne(entityName, inputMap)

`method name: 'findList'`
`type: 'java.util.List'`
`params: [entityName: 'String', inputMap: 'java.util.Map']`
Helping method to find many GenericValue in the database. Used instead of delegator.findList(entityName, inputMap, null, null, null, false)

`method name: 'select'`
`type: 'org.apache.ofbiz.entity.util.EntityQuery'`
`params: [entity: 'java.util.Set']`
Helping method used instead of EntityQuery.use(delegator).select(…)

`method name: 'select', type: 'org.apache.ofbiz.entity.util.EntityQuery', params: [entity: 'String…']`
As above.

`method name: 'from'`
`type: 'org.apache.ofbiz.entity.util.EntityQuery'`
`params: [entity: 'java.lang.Object']`
Helping method used instead of EntityQuery.use(delegator).from(…)

`method name: 'success'`
`type: 'def'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnSuccess(message)

`method name: 'failure'`
`type: 'java.util.Map'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnFailure(message)

`method name: 'error'`
`type: 'def'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnError(message)

`method name: 'logInfo'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logInfo(message, fileName)

`method name: 'logWarning'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logWarning(message, fileName)

`method name: 'logError'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logError(message, fileName)

`method name: 'logVerbose'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logVerbose(message, fileName)

The actual definition of the methods can be found in `` `/framework/service/src/main/java/org/apache/ofbiz/service/engine/GroovyBaseScript.groovy ``, the variables `dctx`, `dispatcher` and `delegator` are set in the file `GroovyEngine.java` which can be found in the same location.

13.3. Services

13.3.1. From MiniLang to Groovy

To see additional examples and finished conversions, which may help with occurring questions, click: OFBIZ-9350 - Deprecate Mini Lang There is a chance that a similar case has already been converted.

When a simple-method ends, it will automatically at least return a success-map.

All the Groovy Services have to return success at least, too.

```
return success()
```

13.3.2. Getting started

MiniLang files consist of services, which, in most cases, implement services.

The get converted to Groovy like the following:

```
<!-- This is MiniLang -->
<simple-method method-name="createProductCategory" short-description="Create an
ProductCategory">
   <!-- Code -->
</simple-method>
```

```
// This is the converted Groovy equivalent
/**
 * Create an ProductCategory
 */
def createProductCategory() {
    // Code
}
```

It will be useful for future developers, and everybody who has to check something in the code, to put at least the short-description as the new Groovydoc. This will hopefully more or less explain, what the method should or shouldn't do. If the short-description isn't helpful enough, feel free complete it.

The structure of if and else in MiniLang is a little different than the one from Groovy or Java and can be a bit confusing when first seen, so here is an example:

```
<if-empty field="parameters.productCategoryId">
    <sequenced-id sequence-name="ProductCategory" field=
"newEntity.productCategoryId"/>
<else>
    <set field="newEntity.productCategoryId" from-field=
"parameters.productCategoryId"/>
    <check-id field="newEntity.productCategoryId"/>
    <check-errors/>
</else>
</if-empty>
```

Notice, that the else always starts before the if-tag is closed, but sometimes isn't indented as one would expect it.

When navigating through bigger `if`-phrases, the navigation itself will be much easier through just clicking in the opening or closing `if`-tag; Eclipse will automatically mark the matching opening or closing `if`-tag for you.

There are two possibilities to initialize a field/variable in Groovy.

1. To define a field/variable with its correct typing
   `String fieldName = "value"``
2. To just "define" a field/variable. The IDE you are working with may not recognize the typing, but OFBiz can work with it:
   `def fieldName = "value"`

13.4. Checking Fields

Minilang	Groovy

`<if-empty field="fieldName"></if-empty>`	`//checks if fieldName is existent and/or empty`
	`if (!fieldName) {}`

`<if-empty field="fieldName.property"></if-empty>`	`// fieldName has to be existent, property doesn't need to`
	`// if known, that property does exist, the ? can be left out`
	`if (!fieldName?.property) {}`
	`// CAUTION: every query like this in Groovy evaluates to a Boolean type`
	`// everything that is empty or false will turn into false:`
	`// null, [], [:], "", false -> false`
	
	`if (UtilValidate.isEmpty(fieldName)) {}`

Minilang	Groovy

`<if>`
`    <condition>`
`        <or>`
`            <if-empty field="field1"/>`
`            <if-empty field="field2"/>`
`        </or>`
`    </condition>`
`    <then>`
`        <!-- code in if -->`
`    </then>`
`    <else>`
`        <!-- code in else -->`
`    </else>`
`</if>`
	`if (!field1 || !field2) {`
	` // code in if`
	`} else {`
	` // code in else`
	`}`

`<if-compare-field`
`field="product.primaryProductCategoryId"`
`to-field="parameters.productCategoryId"`
`operator="equals">`
`    <!-- code -->`
`</if-compare-field>`
	` // this will even work, if product is`
	`not existent or null`
	`if (UtilValidate.areEqual(product`
	`?.primaryProductCategoryId, parameters`
	`.productCategoryId)) {`
	`    // code`
	`}`

`<if-instance-of`
`field="parameters.categories"`
`class="java.util.List"></if-instance-of>`
	`if (parameters.categories instanceof`
	`java.util.List) {}`

13.5. Setting Fields

Minilang	Groovy

`<set field="fieldName" value="value"/>`	` // if fieldName is not initialized`
	`String fieldName = "value"`
	` // if fieldName is initialized`
	`fieldName = "value"`

Minilang	Groovy

`<set field="otherFieldName.property"`
`value="value"/>`
`<set`
`field="otherFieldName.otherProperty"`
`value="true" type="Boolean"/>`
`<set`
`field="otherFieldName.otherProperty"`
`from-field="parameters.property/>`
	` // if otherFieldName is not yet`
	`initialized, you have to do it first`
	` // MiniLang does that automatically`
	`Map otherFieldName = [:] // empty Map`
	` // now put the values in`
	`otherFieldName = [`
	`    property: "value",`
	`    otherProperty: true`
	`]`
	` // or the less efficient way`
	`otherFieldName.property = "value"`
	`otherFieldName.otherProperty = true`
	
	` // it is possible to put different`
	`values in later:`
	`otherFieldName.property = parameters`
	`.property`

`<set field="thisFieldName"`
`value="${groovy: []}" type="List"/>`
	` // this is easier in Groovy`
	`List thisFieldName = []`

`<property-to-field`
`resource="CommonUiLabels"`
`property="CommonGenericPermissionError"`
`field="failMessage"/>`
`<!-- there are different cases of this,`
`which are not distinguished in MiniLang`
`-->`
`<property-to-field`
`resource="general.properties"`
`property="currency.uom.id.default"`
`field="parameters.rateCurrencyUomId"/>`
	`String failMessage = UtilProperties`
	`.getMessage("CommonUiLabels",`
	`"CommonGenericPermissionError",`
	`parameters.locale)`
	` // in Groovy there can be a difference`
	`for the second case`
	`parameters.rateCurrencyUomId =`
	`UtilProperties.getPropertyValue('general`
	`.properties', 'currency.uom.id.default')`

`<clear-field`
`field="product.primaryProductCategoryId"`
`/>`
	`product.primaryProductCategoryId = null`

13.6. Starting Services

Minilang	Groovy
`<set field="relatedCategoryContext.parentProductCategoryId"  from-field="defaultTopCategoryId"/>` `<call-service service-name="getRelatedCategories" in-map-name="relatedCategoryContext">` `    <result-to-field result-name="categories" field= "resCategories"/>` `</call-service>`	`def relatedCategoryContext = [parentProductCategoryId: defaultTopCategoryId]` `def serviceResult = run service: "getRelatedCategoryies", with: relatedCategoryContext` `def resCategories = serviceResult .categories` ` // if it is not too confusing to read you can leave out the extra variable` `run service: "getRelatedCategoryies", with: [parentProductCategoryId: defaultTopCategoryId]`
`<set-service-fields service-name="productCategoryGenericPermission" map="parameters" to-map="productCategoryGenericPermissionMap"/>` `<call-service service-name="productCategoryGenericPermission" in-map-name="productCategoryGenericPermissionMap">` `    <results-to-map map-name="genericResult"/>` `</call-service>`	` // instead of setting the service fields from parameters, it is possible to run the service with the parameters map` `Map genericResult = run service: "productCategoryGenericPermission", with: parameters`

13.7. Preparing Service Results

Minilang	Groovy
`<field-to-result field="fieldBudgetId" result-name="budgetId"/>`	` // MiniLang knows this implicitly` `def result = success()` `result.budgetId = fieldBudgetId` `return result`

13.8. Database Communication

Minilang	Groovy
`<make-value entity-name="FinAccountTrans" value-field="newEntity"/>` `<set-nonpk-fields map="parameters" value-field="newEntity"/>` `<set-pk-fields map="parameters" value-field="newEntity"/>`	`// this is the easy way` `GenericValue newEntity = makeValue("FinAccountTrans", parameters)` `// this is also possible` `GenericValue newEntity = makeValue("FinAccountTrans")` `newEntity.setPKFields(parameters)` `newEntity.setNonPKFields(parameters)`
`<entity-and entity-name="BudgetStatus" list="budgetStatuses">` `    <field-map field-name="budgetId" from-field="parameters.budgetId"/>` `    <order-by field-name="-statusDate"/>` `</entity-and>`	`// this can also be done in one line, but it can easily become unreadable` `def budgetStatuses = from("BudgetStatus")` `    .where("budgetId", paramters.budgetId)` `    .orderBy("-statusDate")` `    .queryList()`
`<entity-one entity-name="StatusValidChange" value-field="statusValidChange">` `    <field-map field-name="statusId" from-field="budgetStatus.statusId"/>` `    <field-map field-name="statusIdTo" from-field="parameters.statusId"/>` `</entity-one>` `<!-- entity-one can be called without child elements, too -->` `<entity-one entity-name="Product" value-field="product" auto-field-map="true"/>`	`// MiniLang has false set for useCache as the default value` `statusValidChange = findOne("StatusValidChange", [statusId: budgetStatus.statusId, statusIdTo: parameters.statusId], false)` `// this is also possible` `statusValidChange = from("StatusValidChange")` `    .where("statusId", budgetStatus.statusId, "statusIdTo", parameters.statusId)` `    .queryOne()` `// if there are no child elements, this can be used` `GenericValue product = from("Product").where(parameters).queryOne()`

Minilang	**Groovy**

```
<find-by-primary-key entity-
name="ProductCategoryMember"
map="lookupPKMap" value-
field="lookedUpValue"/>
```

```
GenericValue lookedUpValue = findOne
("ProductCategoryMember", lookupPKMap,
false)
 // this is also possible
lookedUpValue = from
("ProductCategoryRole")
    .where(lookupPKMap)
    .queryOne()
```

```
<entity-condition entity-
name="ProductCategoryContentAndInfo"
list="productCategoryContentAndInfoList"
filter-by-date="true" use-cache="true">
    <condition-list combine="and">
        <condition-expr field-
name="productCategoryId" from-
field="productCategoryList.productCatego
ryId"/>
        <condition-expr field-
name="prodCatContentTypeId"
value="ALTERNATIVE_URL"/>
    </condition-list>
    <order-by field-name="-fromDate"/>
</entity-condition>
<!-- entity-condition can also be used
with the "or" operator -->
<entity-condition entity-
name="ProdCatalogCategory"
list="prodCatalogCategoryList" 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>
```

```
 // the Groovy methods use the "and" and
"equals" operator as default values
List productCategoryContentAndInfoList =
from("ProductCategoryContentAndInfo")
    .where("productCategoryId",
productCategoryList.productCategoryId,
"prodCatContentTypeId",
"ALTERNATIVE_URL")
    .cache().orderBy("-fromDate")
    .filterByDate()
    .queryList()
 // with the use of the "or" operator
you have to build your condition like
this
EntityCondition condition =
EntityCondition.makeCondition([
    EntityCondition.makeCondition([
        EntityCondition.makeCondition
("prodCatalogCategoryTypeId",
"PCCT_VIEW_ALLW"),
        EntityCondition.makeCondition
("prodCatalogCategoryTypeId",
"PCCT_PURCH_ALLW")
    ], EntityOperator.OR),
    EntityCondition.makeCondition
("productCategoryId", parameters
.productCategoryId)
])
List prodCatalogCategoryList = from
("ProdCatalogCategory").where(condition)
.filterByDate().queryList()
```

Minilang	Groovy
`<make-value entity-name="FinAccountTrans" value-field="newEntity"/>` `<set-nonpk-fields map="parameters" value-field="newEntity"/>` `<!-- In this case multiple fields of the GenericValue are set -->` `<make-value entity-name="ProductCategoryRollup" value-field="newLimitRollup"/>` `<set field="newLimitRollup.productCategoryId" from-field= "newEntity.productCategoryId"/>` `<set field="newLimitRollup.parentProductCategoryId" from-field="productCategoryRole.productCategoryId"/>` `<set field="newLimitRollup.fromDate" from-field="nowTimestamp"/>`	`def newEntity = makeValue("FinAccountTrans", parameters)` `// you can set multiple fields of a GenericValue like this` `def newLimitRollup = makeValue("ProductCategoryRollup", [` `productCategoryId: newEntity.productCategoryId,` `parentProductCategoryId: productCategoryRole.productCategoryId,` `fromDate: nowTimestamp` `])`
`<set field="statusValidChange.prop" value="value"/>`	`statusValidChange.prop = "value"`
`<create-value value-field="newEntity"/>`	`newEntity.create()`
`<store-value value-field="newEntity"/>` `<store-list list="listToStore"/>`	`newEntity.store()` `delegator.storeAll(listToStore)`
`<clone-value value-field="productCategoryMember" new-value-field="newProductCategoryMember"/>`	`def newProductCategoryMember = productCategoryMember.clone()`
`<remove-value value-field="lookedUpValue"/>`	`lookedUpValue.remove()`

Minilang	Groovy
`<sequenced-id sequence-name="ProductCategory" field="newEntity.productCategoryId"/>`	`newEntity.productCategoryId = delegator.getNextSeqId("ProductCategory")`
`<check-id field="newEntity.productCategoryId"/>`	`UtilValidate.checkValidDatabaseId(newEntity.productCategoryId)`

`<make-next-seq-id value-field="newEntity" seq-field-name="linkSeqId"/>`	`delegator.setNextSubSeqId(newEntity, "linkSeqId", 5, 1)`
	` // the numbers 5 and 1 are used in the Java implementation of the MiniLang method`
	` // and can also be found as the default values in the MiniLang documentation`

13.9. Permissions

To also check for admin-permissions, this method has to be used: `hasEntityPermission(permission, action, userLogin)`

If the method is used with wildcards, it is important to not forget the underscore, which comes before the parameter action!

Minilang	Groovy

`<check-permission permission="CATALOG" action="_CREATE">`
`    <alt-permission permission="CATALOG_ROLE" action="_CREATE"/>`
`    <fail-property resource="ProductUiLabels" property="ProductCatalogCreatePermissionError"/>`
`</check-permission>`
`<check-errors/>`
	`if (!(security.hasEntityPermission("CATALOG", "_CREATE", parameters.userLogin)`
	`    || security.hasEntityPermission("CATALOG_ROLE", "_CREATE", parameters.userLogin))) {`
	`    return error(UtilProperties.getMessage("ProductUiLabels", "ProductCatalogCreatePermissionError", parameters.locale))`
	`}`

Minilang	Groovy
`<set field="hasCreatePermission" value="false" type="Boolean"/>` `<if-has-permission permission="${primaryPermission}" action="${mainAction}">` `    <set field="hasCreatePermission" value="true" type="Boolean"/>` `</if-has-permission>`	`// this will automatically be set to false if the user doesn't have the permission` `def hasCreatePermission = security .hasEntityPermission(primaryPermission, "_${mainAction}", parameters.userLogin)`

13.10. Timestamp And System Time

The first two simple-method are deprecated; the third method should have been used instead.

Minilang	Groovy
`<now-timestamp field="nowTimestamp"/>`	`Timestamp nowTimestamp = UtilDateTime .nowTimestamp()`
`<now-date-to-env field="nowDate"/>`	`Timestamp nowDate = UtilDateTime .nowTimestamp()`
`<!-- this method also has the parameter "type", which is set to 'java.sql.timestamp' as default -->` `<now field="fooNow"/>`	`Timestamp fooNow = UtilDateTime .nowTimestamp()`
`<if-compare-field field="productCategoryMember.thruDate" to-field="expireTimestamp" operator="less" type="Timestamp">` `    <!-- code -->` `</if-compare-field>`	`Timestamp thruDate = productCategoryMember.thruDate` `if (thruDate && thruDate.before (expireTimestamp)) {` `    // code` `}`

13.11. Logging

Since all of the log methods are know to the Groovy Language, it is possible to just nearly use them as they are in MiniLang.
For further explanation, here are some examples:

Minilang	Groovy
`<log level="verbose" message="Permission check failed, user does not have permission"/>`	`logVerbose("Permission check failed, user does not have the correct permission.")`
`<log level="info" message="Applying feature [${productFeatureId}] of type [${productFeatureTypeId}] to product [${productId}]"/>`	`logInfo("Applying feature [${productFeatureId}] of type [${productFeatureTypeId}] to product [${productId}]")`

13.12. General

Minilang	Groovy
`<call-simple-method method-name="checkCategoryRelatedPermission"/>` `<check-errors/>`	`// simple-methods inside of classes, as long as they are not services, will be called like normal methods` `Map res = checkCategoryRelatedPermission("updateProductCategory", "UPDATE", null, null)` `if (!ServiceUtil.isSuccess(res)) {` `return res` `}`
`<iterate list="subCategories" entry="subCategory">` `<!-- code -->` `</iterate>`	`for (def subCategory : subCategories) {` `// code` `}` `subCategories.each { subCategory ->` `// code` `}`

Minilang | **Groovy**

```
<iterate-map
map="parameters.productFeatureIdByType"
key="productFeatureTypeId"
value="productFeatureId">
    <!-- in here something should happen
with value and key -->
</iterate-map>
```

```
for (Map entry : parameters
.productFeatureIdByType.entrySet()) {
    def productFeatureTypeId = entry
.getKey()
    def productFeatureId = entry
.getValue()
    // in here something should happen
with value and key
}
```

```
<if>
    <condition>
        <not>
            <or>
                <if-has-permission
permission="CATALOG"
action="_${checkAction}"/>
                <and>
                    <if-has-permission
permission="CATALOG_ROLE"
action="_${checkAction}"/>
                    <not><if-empty
field="roleCategories"/></not>
                </and>
            </or>
        </not>
    </condition>
    <then>
        <!-- code -->
    </then>
</if>
```

```
if (!security.hasEntityPermission
("CATALOG", "_${checkAction}",
parameters.userLogin)
    && !(security.hasEntityPermission
("CATALOG_ROLE", "_${checkAction}",
parameters.userLogin)
    && roleCategories)) {
    // code
}
```

```
<set field="validDate" from-
field="parameters.validDate"/>
<if-not-empty field="validDate">
    <filter-list-by-date
list="productCategoryMembers" valid-
date="validDate"/>
</if-not-empty>
```

```
def query = from(
"ProductCategoryMember").where("productC
ategoryId", parameters.
productCategoryId)
if (parameters.validDate) {
    query.filterByDate()
}
List productCategoryMembers = query
.queryList()
```

Minilang	Groovy

`<order-map-list list="productsList">`
`    <order-by field-name="sequenceNum"/>`
`</order-map-list>`
	`productsList = EntityUtil.orderBy`
	`(productsList, ["sequenceNum"])`

13.13. Where to find MiniLang implementation

If you find yourself in a position, where you don't know how to convert a certain tag from MiniLang to Groovy, you can always check the Java implementation of the MiniLang method.
All of the methods have an existing Java implementation and you can find all of them in this folder: `/ofbiz/trunk/framework/minilang/src/main/java/org/apache/ofbiz/minilang/method`

The interesting part of this implementation is the method `exec()`, which actually runs the MiniLang tag.
The tag `<remove-by-and>` for example is realized using this part of code here:

```
@Override

public boolean exec(MethodContext methodContext) throws MiniLangException {
    @Deprecated
    String entityName = entityNameFse.expandString(methodContext.getEnvMap());
    if (entityName.isEmpty()) {
        throw new MiniLangRuntimeException("Entity name not found.", this);
    }
    try {
        Delegator delegator = getDelegator(methodContext);
        delegator.removeByAnd(entityName, mapFma.get(methodContext.getEnvMap()));
    } catch (GenericEntityException e) {
        String errMsg = "Exception thrown while removing entities: " + e.getMessage();
        Debug.logWarning(e, errMsg, module);
        simpleMethod.addErrorMessage(methodContext, errMsg);
        return false;
    }
    return true;
}
```

In this you can find one important part of code, which is:

```
delegator.removeByAnd(entityName, mapFma.get(methodContext.getEnvMap()));
```

This tells you, that, if you're trying to convert the tag `<remove-by-and>`, you can use `delegator.removeByAnd()` in Groovy.