ARTIO OB Bridge Documentation

Tomáš Vantuch
Michal Unzeitig

OB Bridge Documentation

ARTIO

Publication date: 5.8.2013
Version: 1.0.1

Abstract

User documentation for ARTIO synchronization (OB Bridge) component for Openbravo ERP!. This document describes the component requirements, installation and usage on both sides of synchronization process, Magento and Openbravo. All main component parts are described in more details, including illustration images and examples. It specifies server and browser requirements and information about environment constraints. The documentation is helpful for both web developers and website administrators / owners.

Table of Contents

Chapter 1. Introduction

OB Bridge, online connector, is a specialized set of software components for Openbravo and Magento for online data synchronization between these systems. It becames required part of ERP Openbravo because covers synchronization of all important entities. Important entities are in our case these business objects:customer with contact information, addresses, categories, products, products attributes, prices and of cource orders with payments, shippings and parcel numbers. The whole synchronization is running on the background and can be completely scheduled, so user doesn't have to spent his time by managing processes over and over again. There is no needed to special settings if component is installed correctly. Data synchronization is running automatically in background. Integration covers synchronization data from Magento to Openbravo and vice versa.

1.1. Technical Requirements

1.1.1. Server Requirements

There are two server sites to handle: Openbravo server and Magento server.

The component is compatible and has been tested with:

- Openbravo3 MP22.1
  - Java 1.6.0_45 (x84)
  - PostgreSql 9.2.4
  - Tomcat 6.0
  - Ant 1.7.1
- Magento 1.7.0.2
  - php 5.3.13
  - MySql 5.5.24
  - Apache 2.2.22

1.1.2. Software Requirements

OB Bridge uses XmlRpc client 3.1.3 Java framework from Apache foundation. When using different versions of this framework, proper functionality may not be guaranteed. OB Bridge contains this library in its installation set.

Please note, that you can not use XmlRpc Client framework at the same time without special adjustment. There may exist a conflict between these frameworks that needs to be resolved. All libraries should run in non-conflict mode.

1.1.3. Database Requirements

OB Bridge will do modification in the Openbravo database during installation. Please, ensure that database user has all permissions to execute commands for add new table, column or trigger.

1.1.4. Browser Compatibility

The component behavior is not depend on you web browser choice. So feel free at choosing your favorite one, recomended is Chrome browser

Chapter 2. Openbravo added features

OB Bridge brings additional features to Openbravo beside full integration with Magento. Below is list of the new features, which are now available.

2.1. Category tree

Default Openbravo does not enable to assign one product to the more categories. OB Bridge brings this well know feature from Magento so one product could be placed into several categories now. Whole structure of categories and sub-categories is synchronized to Magento.

2.2. Configurable products

Configurable product is a known term from Magento. OB Bridge brings same functionality into Openbravo. User can make more products with the same attribute set, define for each one an attribute value and create one configurable product based on the same attribute set. As a result you get only one product in your web store that wraps all child-products with unique attribute values.

2.3. Related products

Related products works the same way as in Magento now. There is one product for example LCD monitor and there are two products DVI and HDMI cable as its related products that can be bought with monitor. The same relations and minimum quantity of related product to main product can be defined in Openbravo.

2.4. Product files

This functionality allows to add to product additional files like images, pdfs, docs etc. These files are synchronized to Magento as well.

2.5. Notifications

OB Bridge provides API for other modules to be notified about the paid sales orders. Sales orders are checked periodically using background process and for every paid sales order all the registered listeners from other modules are called.

2.6. Coupon codes

It brings core functionality of Magento into Openbravo as well. User can define speecial product associated to category (probably coupon category), product has its rules (discount amount, counts of usage, etc.). When that product is bougth, the coupon code is generated, sent to the customer via email and sent to Magento as well for synchronization.

2.7. Weditor

Provides a WYSIWYG editor for input fields in Openbravo.

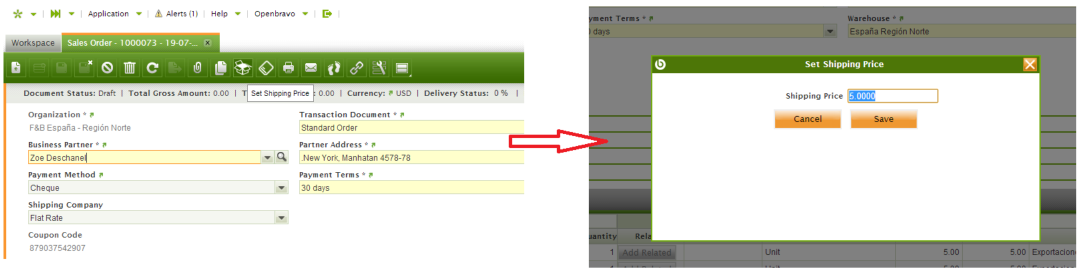
2.8. Order recalculation

Provides API for other modules to be notified about the Recalculate Promotions button in Sales Order window. There's a custom Java process that must be assigned to the Recalculate Promotions button in order to work properly and be notified.

2.9. Adjustable shipping price

Shipping price is defined as another order line with name of shipping company and price that is set for the order. You can set it by click on "Set Shipping Price" and put in required price.

Figure 2.1. Setting the shipping price



Document Status: Draft | Total Gross Amount: 0.00 | T… : 0.00 | Currency: USD | Delivery Status: 0 %

Organization *	F&B España - Región Norte
Transaction Document *	Standard Order
Business Partner *	Zoe Deschanel
Partner Address *	.New York, Manhatan 4578-78
Payment Method	Cheque
Payment Terms *	30 days
Shipping Company	Flat Rate
Coupon Code	879037542907
Warehouse *	España Región Norte
Shipping Price	5.0000

uantity	Rela				
1	Add Related	Unit	5.00	5.00	Exportacion

Chapter 3. Installation

3.1. Prerequisities

Before installing ARTIO OB Bridge, you need to have:

1. Openbravo 3 MP22.1 and Magento 1.7.0 correctly installed.

Make sure that all the needed directories (as requested by Openbravo or Magento) are writable. And database users have all granted permissions.

3.2. Installation

1. into the Magento
   - copy all modules to directory /app/code/local/Artio/...
   - copy all config files to /app/ets/modules/...
   - clean the cache and run
2. into the Openbravo
   - copy all modules to directory /modules/...
   - run commands "ant update.database" and "ant smartbuild" in command line

Please, watch for any error messages generated during installation. If any error appears, the installation might not have been completed correctly.

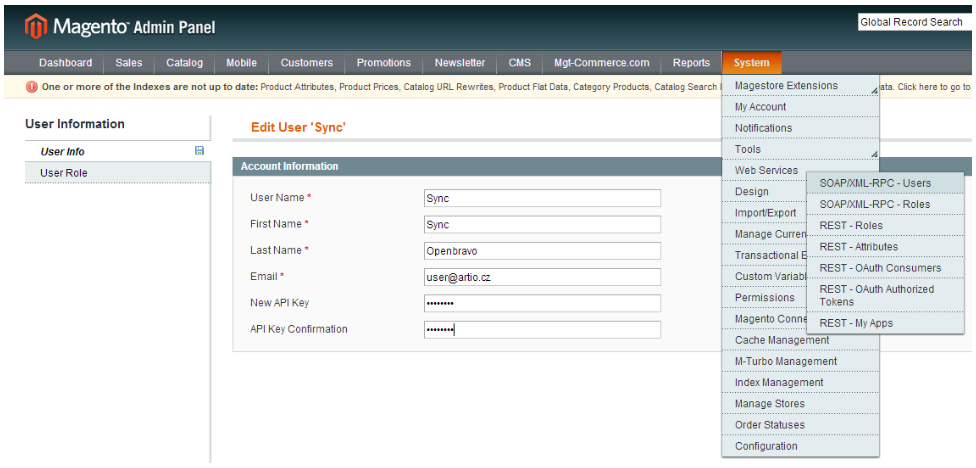
Chapter 4. Administration (backend)

4.1. Magento API user

The main point of communication between Openbravo and Magento is calling Magento web services API from Openbravo. All requests are secured on Magento site by username and password, that must be assigned to our connection. To create our XML-RPC User just follow these steps:

In Admin mode go to System -> Web Services -> SOAP-XML-RPC-Roles and create a role with all permisions. Next go to System -> Web Services -> SOAP-XML-RPC-Users and create an user with assignation of registered role.

Figure 4.1. Creating Magento Xml-rpc user



One or more of the Indexes are not up to date: Product Attributes, Product Prices, Catalog URL Rewrites, Product Flat Data, Category Products, Catalog Search I ata. Click here to go to

User Name *	Sync
First Name *	Sync
Last Name *	Openbravo
Email *	user@artio.cz
New API Key	••••••••
API Key Confirmation	••••••••

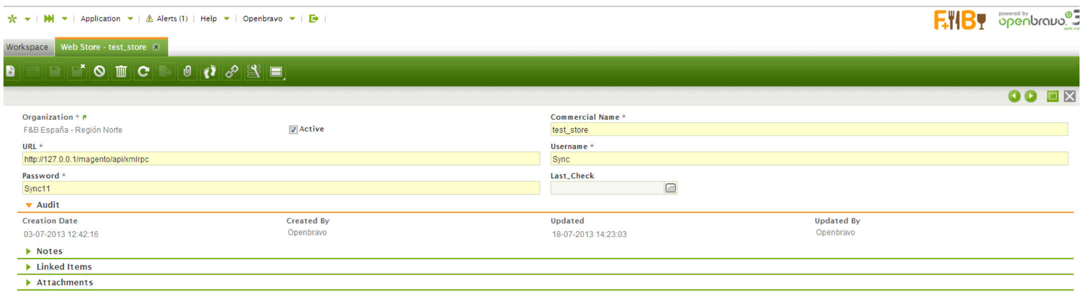
4.2. Openbravo web store configuration

As a next step, we need to configure a web store in our Openbravo installation. Web store is nothig else, just record contains URL of Magento API and username and password for secured connection.

To set Web store just open window "Web store", create a new record and fill all required fields:

- Organization: this organization is assigned to all created entities records by default
- Commercial name: name of your web store - used just as note.
- URL: direct url address to your magento API, you can create it just by http://<address of the store>/api/xmlrpc
- Username: name of the user from magento XML-RPC User registration
- Password: password of the user from magento XML-RPC User registration

Figure 4.2. Web store configuration



Organization * F&B España - Región Norte
☑ Active
Commercial Name * test_store
URL * http://127.0.0.1/magento/api/xmlrpc
Username * Sync
Password * Sync11
Last_Check
Creation Date 03-07-2013 12:42:16
Created By Openbravo
Updated 18-07-2013 14:23:03
Updated By Openbravo

Tip

: There can be only one created web store, if you create more than one, it will be used only the first active one.

4.3. Openbravo process settings

When Openbravo is able to connect to Magento via API, it is possible to define synchronization rules. It means to set value how often must synchronization for particular object occurs. For instance, lets take a customer entity. It can sychronized quite often, regularely about every ten minutes. Other entity, for example customer categories, can be synchronized just in time of created or updated some. :

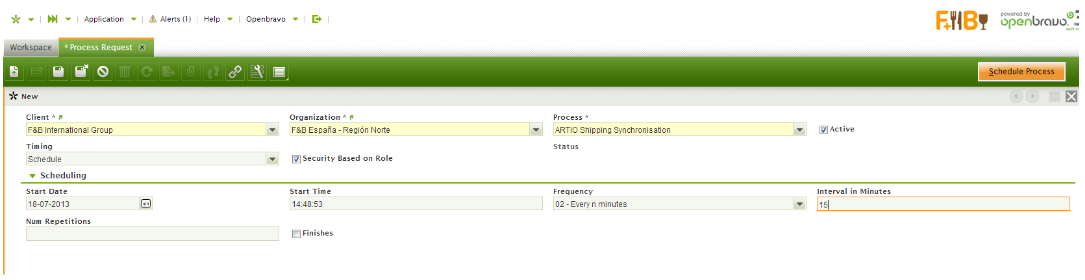
Note

: every synchronization process takes system resources for its running, so take it carefully when you are creating them.

To create new synchronization process, is needed to open "Process Request" window and fill these fields:

- Client: openbravo client, accessible during process from OBContext
- Organization: openbravo organization, accessible during process from OBContext
- Process: registered synchronization process
- Timing: options to run it immediately, scheduled or later in defined time

Figure 4.3. Process Request registration



Client * F&B International Group
Organization * F&B España - Región Norte
Process * ARTIO Shipping Synchronisation
☑ Active
Timing Schedule
☑ Security Based on Role
Status
Start Date 18-07-2013
Start Time 14:48:53
Frequency 02 - Every n minutes
Interval in Minutes 15
Num Repetitions
☐ Finishes

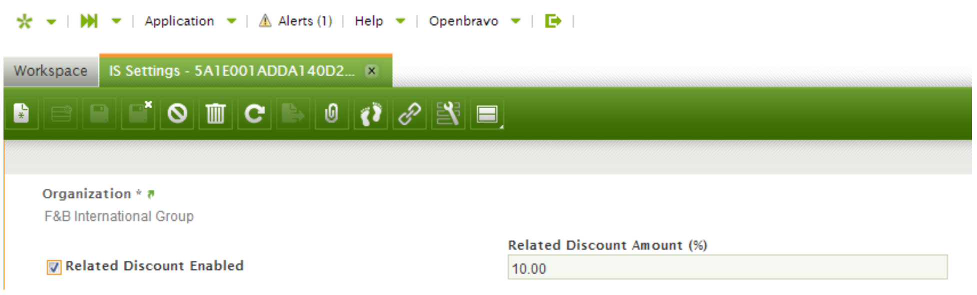
4.4. Openbravo IS settings

In case of using some constants with defined values that needs to be the same in both systems, you can set them here. In Openbravo, there is a database table AS_Settings handled from window IS Settings. Synchronization is supported by ISSettingsProcess, that takes needed attributes and their values to save them in magento. Only these two are now important to know:

4.4.1. Related products discount

OB Bridge contains support for related products and this is a place where you can define a special discount for all related producs. This feature supports to set a discount for specific amount of child-products, which are related to the main product, if child-producs are bough at one with main product in one order. If the discount is set to zero, there is no discount applied.

Figure 4.4. Related product discount



Organization *
F&B International Group

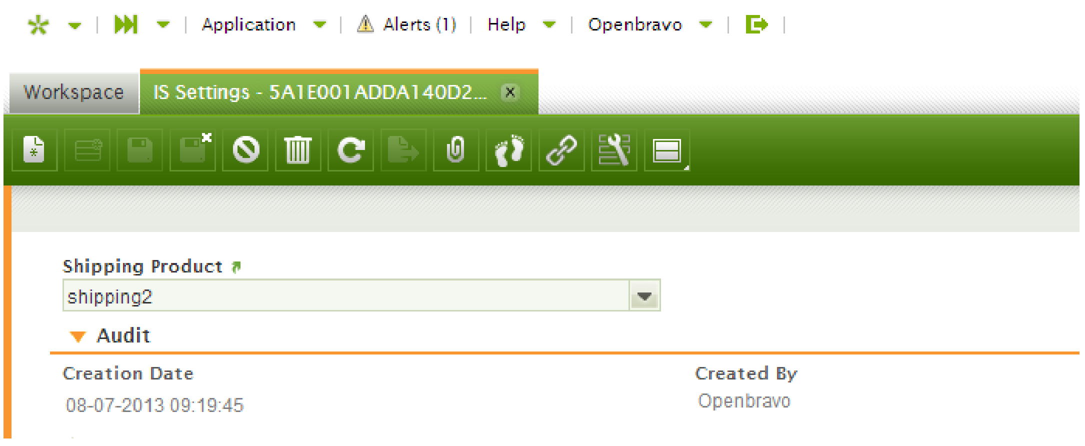
[✓] Related Discount Enabled

Related Discount Amount (%)
10.00

4.4.2. Shipping product

This is very important part of handling shipping price to each order. If you create an order at Magento site, the shipping price is adjustable by settings. Openbravo has an option to define a shipping price as was written in chapter 2. To allow this functionality, is needed to create a simple product, name it as you wish, with no attrbutes and no prices and in the window IS Settings assign it as a shipping product. Thanks to this, it will be assigned to each order as another order line that will define a shipping price.

Figure 4.5. Shipping product



Shipping Product
shipping2

Audit

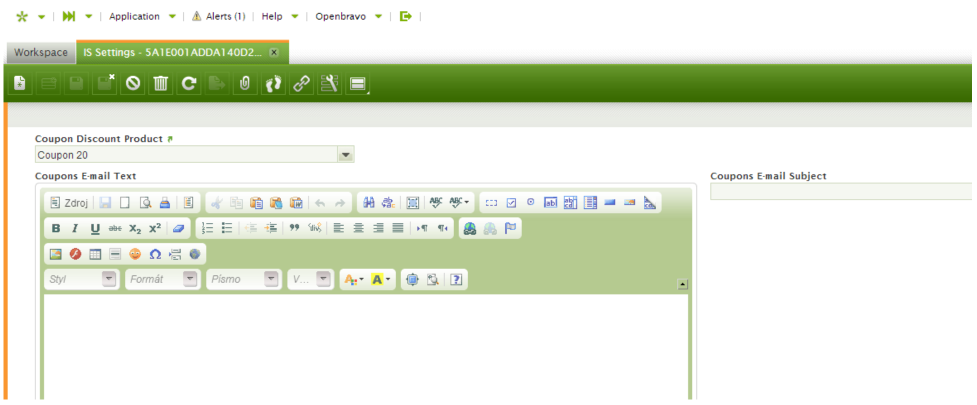
Creation Date
08-07-2013 09:19:45

Created By
Openbravo

4.4.3. Coupon discount product

Customer can buy voucher as virtual product. It means that voucher with uniqe code is generated and could be used as gift. Checkout page of Magneto has a field for enter voucher code. Magento validate if voucher code is corrent and apply discount. Order contains one more line with discount value. Discounted order is synchronized into Openbravo.

Figure 4.6. Coupon code settings



Coupon Discount Product: Coupon 20

Coupons E-mail Text

Coupons E-mail Subject

Chapter 5. Synchronization processes

5.1. Customer Category

In default, this process handles the synchronization of customer categories from Openbravo to Magento. If category is named "Supplier Services" or "Supplier", could be changed in the configuration, the process will assign no tax to them.

Name: Customer Category Synchronization

Way: One way, from Openbravo to Magento

Tables:

- Openbravo:
  - C_Bp_Group
- Magento:
  - customer_group

5.2. Customer

This process reflects all updates on customer entity, include its addresses and contact information on both sites of communication.

Name: Customer Synchronisation

Way: Two way, from Openbravo to Magento and vice versa.

Tables:

- Openbravo:
  - C_Bpartner
  - Ad_User
  - C_BPartner_Location
- Mageno:
  - customer_entity*
  - customer_address*

Notice: Two way synchronization can cause conflicts. That means the situatition when on both sites a newly updated record appears and it's ready to synchronize. This case is solved in this way: updated record from Magento will replace the changes in Openbravo. Message about the potential conflict is written into log file.

*includes all derivated tables by attribute types (_int, _date, _text, _varchar)

5.3. Product Category

Product categories, its names, descriptions, places in a category tree are moved from Openbravo to Magento by this process. It can reflect also changes in category tree by moving categories on Magento site.

Name: Product Category Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_Product_Category
  - AD_Tree
  - AD_TreeNode
- Mageno:
  - catalog_category_entity*
  - catalog_category_flat_store

*includes all derivated tables by attribute types (_int, _date, _text, _varchar)

5.4. Attribute and Attribute Set

Product can be more specificaly defined by its attributes wapped in attribute set. They can be created and managed on Openbravo and Magento. This process allows user to define them only in Openbravo with effects on Magento side.

Name: ARTIO AttributeSet Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_AttributeSet
  - M_Attribute
  - M_AttributeUse
- Mageno:

  - eav_attribute
  - eav_attribute_set

Notice: This process creates only attribute sets and attributes. All attributes are created as type "select" and values for them are sending by product process directly to the saved product.

5.5. Product

Product itself is moved from Openbravo to Magento by this process. Synchronization includes information about prices, descriptions, meta-information, added attributes by attribute set, tax category and visibility.

Name: Product Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_Product
  - M_AttributeValue
  - M_ProductPrice
  - AEC_Product_Cat_Xref
- Mageno:
  - catalog_product_entity*
  - catalog_product_index_price
  - catalog_product_link
  - catalog_product_link_attribute*

**includes all derivated tables by attribute types (_int, _date, _text, _varchar)*

5.6. Product Relations

Product Relations is a binding between two products that gives one to position of related to second. This relation is moved to Magento by this process on already synchronized products.

Name: Product Relations Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - ARP_Related_Product

  - M_Product
- Mageno:
  - catalog_product_relation

5.7. Product Image

Process ensures synchronization images for products from Openbravo to Magento.

Name: Product Image Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_Product
  - APAF_File
- Mageno:
  - catalog_product_entity_media_gallery
  - catalog_product_entity_media_gallery_value

Notice: This process moves only information about picture (name, address, type), not picture itself. This means that user have to physicaly move his images to required directory and information like name, type, address are moved by this process into Magento database. It is possible to use rsync to sychronize product images.

5.8. Product Stock Data

Products stock data, information about avalable counts of each product in warehouse is by this process moved into Magento. It's very simple process that moves just a little data so it doesn't take a lot of system resources. By this way it can be scheduled in shorter iterations.

Name: Stock Data Synchronisation

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_InventoryLine
- Mageno:
  - cataloginventory_stock_item
  - cataloginventory_stock_status

5.9. Coupon Rule

Coupon rule defines conditions for getting discount at the cart/checkout. Scenario how this whole process wiht synchronization works is described in chapter 2.6. This process has as a task to move Openbravo coupon products as coupon rules to Magento.

Name: Coupon Rules Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_Product
  - M_Product_Category
- Mageno:
  - salesrule
  - salesrule_website

Notice: This process creates products too. It is purchasable product that generates coupon code to customer.

5.10. Coupon Code

After coupon product is bought and paid, on Openbravo site is created coupon code that is synchronized to Magento by this process. It allowes customer to use his coupon on eshop.

Name: Coupon Code Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - ACPN_Coupon_Code
- Mageno:
  - salesrule_coupon
  - salesrule_coupon_usage

Notice: After using coupon on order, discount is show and moved back with code to Openbravo, so there is no option to use code more than once.

5.11. Order

Ther is this process that handles moving records from Magento eshop to Openbravo ERP and back to synchronize new or updated order. Customer can see on Magento website own orders and made changes. The order entity record in our terminilogy means an order with all order items, customer with his billing and shipping address and shipping price and discounts too.

Name: Order Synchronization

Way: Two way, from Openbravo to Magento and back.

Tables:

- Openbravo:
  - C_Order
  - C_Orderline
  - C_OrderlineTax
  - C_BPartner
  - C_BPartner_Location
- Mageno:
  - sales_flat_quote*
  - sales_flat_quote_address*
  - sales_flat_quote_item*
  - sales_flat_quote_item_option*
  - sales_flat_quote_payment*
  - sales_flat_quote_shipping_rate*
  - sales_flat_order
  - sales_flat_order_item
  - sales_flat_order_address
  - sales_flat_order_grid

Notice 1: Order can not be updated at Magento site, so when update happens on Openbravo site, process will reorder (cancel old and create new) the order on Magento site.

Notice 2: Order has its specific order number, it is created defaultly in Magento and Openbravo too. As option to show to the customer the right order number, there is in Magento created table order_number where are stored all that number during synchronization.

5.12. Payment

Payment of order is a little more complicated issue, and by this way is handled by two processes. There is to solve creating only one invoice, synchronize payments both ways and booking orders too. So let's split this chapter to make it simple.

5.12.1. From Magento to Openbravo

There is option when order is payd directly after is was created. It was payd by credit card or paypal... The main point is, that on Magento site was created invoice base on payment. These invoice is better not to show to customer because now it's only record of payment for synchronization process. The process takes all new invoices on Magento site and by them it will book orders assigned to them on Openbravo and by the amount there will be created payment. Now Orders are paid on Openbravo and ready to continue by shipping and invoicing...

Name: Payment Synchronization

Way: One way, from Magento to Openbravo.

Tables:

- Openbravo:
  - FIN_Payment
  - FIN_Payment_Detail
  - FIN_Payment_Schedule
  - FIN_Payment_ScheduleDetail
- Mageno:
  - sales_flat_invoice
  - sales_flat_invoice_grid
  - sales_flat_invoice_item
  - sales_flat_order_payment

5.12.2. From Openbravo to Magento

In other way, when payment was placed on Openbravo site and there was created invoice of the order, the invoice can be sended to customer by mail and by synchronize process Order in Magento site will be invoiced. That will prevet to show paid order as not paid to customer.

Name: ARTIO Invoice Synchronisation

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - C_Invoice
  - C_InvoiceLine
  - FIN_Payment_Schedule

  - FIN_Payment_ScheduleDetail
- Mageno:
  - sales_flat_invoice
  - sales_flat_invoice_grid
  - sales_flat_invoice_item
  - sales_flat_order_payment

5.13. Shipping

To synchronize record that point on order what is and what is not shipped, there is this process. A part of that is sending tracking numbers of packages to order on Magento site, to notice his future owner.

Name: ARTIO Shipping Synchronisation

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - M_InOut
  - M_InOutLine
  - Aship_Parc_No
- Mageno:
  - sales_flat_shipment
  - sales_flat_shipment_item
  - sales_flat_shipment_track

Notice: When order is marked as "can be shipped separately", there is option to send not all order items but just available product. In this case there will be synchronized this partly shipping.

5.14. IS Settings

Settings presented in previous chapter (4) can be importnat in Magento too. For example the related product discount is moved as constant into Magento to the tabel core_config_data with record path "sales/related_discount/discount_in_percent".

Name: ARTIO IS Settings Synchronization

Way: One way, from Openbravo to Magento.

Tables:

- Openbravo:
  - AS_Settings
- Mageno:
  - core_config_data

5.15. Delete

When some entity record is deleted on Openbravo and we need to delete him on the Magento site, we need to use this synchronization process.

Name: ARTIO Delete Process

Way: One way, from Openbravo to Magento.

Notice: This is ready for enetities : customer, product, product category and order (not booked or paid).

Chapter 6. Important knowledges before start

Notice about Openbravo Organization settings : Most processes are one way synchronization from Openbravo to Magento. But processes that saves new data from Magento to Openbravo needs to know what is the right Organization for created entity record (if you have them more than one). For example if a newly created order has assigned wrong Organization (different as product in that order or tax category or tax rate,...) there is no option to correctly count taxes on this order. To all new orders is assigned Organization that you have setted in Web Store configuration from chapter 4, section 2. And to all order lines, shipping lines and payments record will be assigned the same one. For new customer in Openbravo site is that the same situation, the organisation assigned to him is taken from web store config too.

Notice about payment methods : In default build of OB Bridge there is support for only one kind of payment method to synchronize for order. It is payment by check. There are two reasons, why only this one. At first is this only one that exists in clean magento and openbravo setup defaultly. And as second reason, that any other company will add on both sides the payment methods that they are using. How to add them programmatically is described in next chapter as tutorial.

Notice about shipping methods : This is the same story as in previous paragraph. The only one shipping method that is availible in this build of OB Bridge to synchronize is FLAT_RATE. The reasons are the same as in previous and tutorial for adding new one is ready too.

Chapter 7. How to extend

7.1. MagentoSyncProcess

In time, when you need to create your own synchronize process, you can just implement the abstract class net.artio.magento.MagentoSyncProcess.java and it will give you a set of method to implement.

- String getItemPrintName(HashMap<String, Object> itemData) - In case of logging, this method can return user friendly item print name of entity record
- boolean initialize(AMSPWebStore webStore, XmlRpcClient client, String sessionId) - to init process by saving global values and return true if everything is ok.
- String getChangedItemsMethod() - Returns name of magento API method that returns a list of newest entity records (identifiers).
- String getItemListMethod() - Returns name of magento API method that returns an info of newest entity records (detail view).
- BaseOBObject synchronizeItem(Object itemData) - to provide saving new or updated founded entity.
- String getName() - This method holds and returns process name
- Vector<Object> getChangedItemsArguments() - Sets arguments for magento API. Like lastTime of sync (to get an updated records from that time)
- List<Object> getItemsFromOB() - This is called in iterations so the best way is to returns entity records in bashes.
- String getSyncMagentoMethod() - returns name of magento API method, that will handle our sended data
- Object toItemData(Object entity) - Convert Openbravo class model (can be more than one class type) to hashMap where keys are defined in Magento API.
- void resolveKeys(Object[] keys) - to save magento indentifiers as result of synchronization.
- void actualizeSyncData() - to store time when process was runned at last.
- String getId(Object object) - returns openbravo identifier of the entity.
- Object getItemsToFilter(Object object) - returns ids or conditions to get details of entity record.
- void refreshMagSyncTime() - Entity can have a record in AMSP_Data table where is stored last time of calling of magento API for that entity.
- Object[] saveKeys(Object[] items) - Keys that come from magento in MAG -> OB way, can be stored to prevent lost of them during some failure.
- void handleException(Exception e) - to save exception to other log file we can use this.

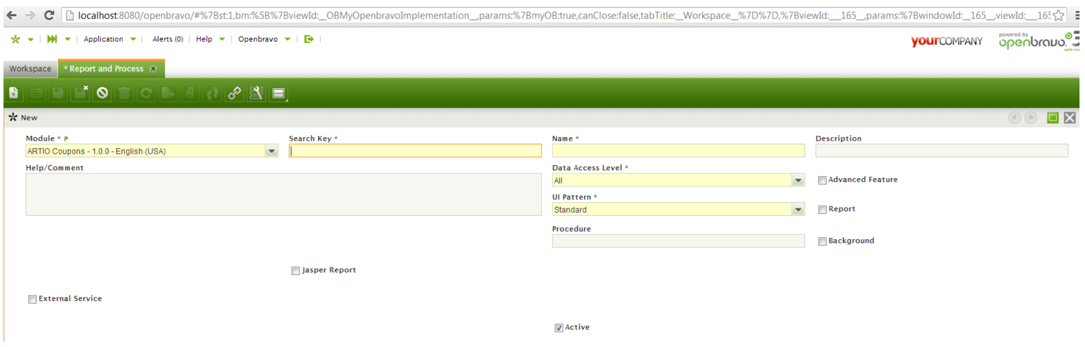
Complete documentation describes implemented methods, inputs and outputs is in java doc.

To create process that will synchronize entity records one way, you do not have to implement all of this methods. In every case, the method initialize have to return true, so that's the firts thing you have to check. To implement GetName and getItemPrintName is good for logging.

- for Openbravo -> Magento way just implement : getItemsFromOB, getSyncMagentoMethod, toItemData, resolveKeys, actualizeSyncData and getId. (for example AttributeSetProcess.java)
- for Magento -> Openbravo way just implement : getChangedItemsMethod, getItemList-Method, synchronizeItem, getChangedItemsArguments, getId, getItemsToFilter. (for example PaymentProcess.java)
- for both way synchronization you need to implement all the methods. (for example OrderProcess.java)

After your process is ready, you need to register it into Openbravo. Log in as System Administrator, go to window Report and process and create a new process record.

Figure 7.1. Report and Process window



localhost:8080/openbravo/#%7Bst:1,bm:%5B%7BviewId:_OBMyOpenbravoImplementation_,params:%7BmyOB:true,canClose:false,tabTitle:_Workspace_%7D%7D,%7BviewId:__165__params:%7BwindowId:__165__viewId:__16

Module *: ARTIO Coupons - 1.0.0 - English (USA)
Search Key *:
Name *:
Description:
Help/Comment:
Data Access Level *: All
UI Pattern *: Standard
Procedure:
☐ Advanced Feature
☐ Report
☐ Background
☐ Jasper Report
☐ External Service
☑ Active

7.2. Triggers on Openbravo

Triggers are one of the best and maybe the fastest way how to handle changes on entity records. How to create own trigger is described in this tutorial: http://wiki.openbravo.com/wiki/ERP_2.50:Developers_Guide/How_To_Create_a_Trigger

```
-- Function: amsp_m_product_trg()

-- DROP FUNCTION amsp_m_product_trg();

CREATE OR REPLACE FUNCTION amsp_m_product_trg()
  RETURNS trigger AS
$BODY$ DECLARE

 BEGIN
 SELECT COUNT(*) INTO v_count FROM AMSP_Process;
 RAISE NOTICE '%',v_count;
 IF v_count > 0 THEN

  RETURN NEW;
 END IF;

 IF TG_OP = 'UPDATE' THEN
  BEGIN
  IF(NEW.EM_AMSP_MAGENTO_KEY = OLD.EM_AMSP_MAGENTO_KEY
   AND NEW.EM_AMSP_SYNC_AT = OLD.EM_AMSP_SYNC_AT)
  THEN
   NEW.EM_AMSP_SYNC_AT := null;
```

```
    RETURN NEW;
   END IF;
 RETURN NEW;
 END;
 END IF;
    IF TG_OP = 'DELETE' THEN
 BEGIN
  INSERT INTO AMSP_DELETE VALUES(get_uuid(), '0', '0', 'Y', TO_DATE(NOW()), '0', TO_DATE(NOW()),
 '0', 'product', OLD.value);

 RETURN OLD;
 END;
    ELSE
    RETURN NEW;
    END IF;

END
; $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION amsp_m_product_trg()
  OWNER TO tad;
```

Each entity where we need synchronize updated records too, have to be marked. As a main synchronization mark is used em_amsp_sync_at column in the Openbravo entity table. When it is empty, the entity record have to be synchronized. When the entity record is updated, this is a right event to clear that column. So in each entity there is a trigger ready to handle update event where is only one simple condition. When there is an update event and the new record has the same em_amsp_magento_key and em_amsp_sync_at (so it was not actualized during synchronization) as the old record, the update belongs to another entity columns and em_amsp_sync_at has to be errased (set to null). After that it will be finded by his process to synchronization.

Some triggers handles delete event too. Entities where delete have to be performed on Magento site too like customer or product just run their trigger on delete event and to table amsp_delete type of entity and its magento identifier* (em_amsp_magento_key => if is null, the entity record was not synchronized and there is no need to synchronize its delete) is stored. and of cource the scheduled Delete process will iterate all records and run Magento API methods to delete them.

*in case of product, there is sku used as identifier

7.3. Extending Magento API

Best way how to start extending API is to read some official tutorial, like this http://www.magentocommerce.com/wiki/doc/webservices-api/custom-api. In the next step is very important to know in what shape we are sending data from Openbravo and in what shape we are waiting for an answer. Let's have a look at this:

- As magento API input all methods used by OB bridge are waiting for an array of objects that consists of two parts. First one is 'ob_id' that is a return mark. The return answer will contains this id and we will know if that object was synchronized correctly. And the second one is 'data' and it's array of data of the sended object. The inner structure of 'data' depends of course on the type of saved entity.

- As an output we are expecting an array of associated arrays (in java hashmaps) where is ob_id (identifier of sended object) as a key and result of operation (mostly magento identifier of that object). If we have magento identifier sended back to openbravo after correct synchronization, we can store it and use it in time of update. Most of the synchronized entities on Openbravo site has added columns:

    - em_amsp_magento_key - identifier from magento for that entity record

- em_amsp_sync_at - timestamp of last synchronization

With knowledge of input and output you can create your own methods that you will call from Openbravo scheduled process.