Price Engine

Introduction

The Price Engine is responsible for calculating all kinds of prices in the shop. It is able e.g. to calculate prices based on imported prices and rules, but also to use the business logic of the ERP system which is connected to the eZ Commerce.

It offers a very flexible way to combine the logic of an ERP system and a local price provider in order to get the best compromise between realtime data and a performant shop. In addition it offers a failover concept in case of the ERP is not available.

> The base entry point for price engine is **ChainPriceService**, which is used to fetch prices.

It determines a chain of Price Providers, which will be responsible for calculating the prices.

It is up to configuration, which set of Price Providers will be used. This is important because depending on the page (product listing, product detail, checkout etc.) different requirements have to be solved:

- on a **product list** a lot of prices might have to be calculated. This might cause problems when using the Business logic of the ERP. In this case a local Price Provider will be fast in order to provide e.g. list prices
- on a **product detail** page the customer expects to get his individual price from the ERP
- in the **basket** the price shall always be provided by the ERP

The ChainPriceService is using *ContextId* (e.g. basket, product_list). For each context a prioritized list of Price Providers can be defined. This concept also allows to define a fallback if e.g. the ERP is not available. The response for a price request contains a source so that the shop can display e.g. a warning if the price and stock is not provided by the ERP but by a fallback price provider.

In addition to prices, the ChainPriceService is able to retrieve stock information since the ERP systems usually provide this information in the price request.

Available price providers

Provider	Logic	Note
Local price provider	A simple one getting the price from the product itself	⚠ Do not use this provider for eZ Commerce. Please use ShopPricdEngine instead!
Shop price provider	A more sophisticated price provider. Offers currency and customer group support	
Remote price engine	Gets prices from the ERP	Advanced version only!

How to get prices from the price engine

This example shows how to get a price from the ERP for a given SKU "D4142".

- First the productNode is fetched by using fetchElementBySku()
- Afterwards the price engine is called using the priceContext "basket".
- The method addPricesToProductNodes() will enrich the ProductNode with customer specific prices and stock infos

For basket a realtime request to the ERP is configured:

Table of contents:

Child Pages:

```
siso_price.default.price_service_chain.basket:
        - siso_price.price_provider.remote
        - siso_price.price_provider.local
```

```
$skuList = array('D4142');
        $catalogService =
$this->getContainer()->get('silver_catalog.data
_provider_service');
        foreach ($skuList as $sku) {
            /** @var $catalogElement
\Silversolutions\Bundle\EshopBundle\Product\Pro
ductNode */
            $catalogElement =
$catalogService->getDataProvider()->fetchElemen
tBySku(
                $sku,
                array()
            );

            if ($catalogElement instanceof
\Silversolutions\Bundle\EshopBundle\Catalog\Cat
alogElement) {
                $productNodeArray[] =
$catalogElement;
            }
        }

        $catalogService =
$this->getContainer()->get('silver_catalog.cata
log_service');

$catalogService->addPricesToProductNodes($produ
ctNodeArray, "EUR", "basket",
            array(1));
        $price =
$productNodeArray[0]->customerPrice->price->pri
ce;
        $isOnStock
=$productNodeArray[0]->stock->isOnStock();
        echo "Price: $price";
        echo "OnStock: ".$isOnStock;
```

If you do not want to fetch a sku using the dataprovider you have to create a **ProductNode manually**:

```
            $price = new
\Siso\Bundle\PriceBundle\Model\Price(
                array(
                    'price' => 12.00,   //
Fallback price
                    'isVatPrice' => true,
                    'vatPercent' => (float)19,
                    'currency' => 'EUR',
                    'source' => 'ERP',
                )
            );
            $priceField = new
\Silversolutions\Bundle\EshopBundle\Content\Fie
lds\PriceField(array('price' => $price));
            $attributes = array(
                'sku'     => 'D4142',
                'vatCode' => 'VATNORMAL',
                'price'   => $priceField
            );
            $catalogElement = new
\Silversolutions\Bundle\EshopBundle\Product\Ord
erableProductNode($attributes, $urlService);


            ## If a variant is used
            $variantCode = "001";
            $variantService =
$this->get('silver_catalog.variant_service');
            $catalogElement =
$variantService->createOrderableProductFromVari
ant($catalogElement, $variantCode);
```

Important terms used in this document

BuyerParty

BuyerParty is party which acquires, or agrees to acquire, ownership (in case of goods), or benefit or usage (in case of services), in exchange for money or other consideration under a contract of sale. Also called purchaser.
Source: http://www.businessdictionary.com/definition/buyer.html

ContextId

ContextId is a string value which is used to describe in which context the requested price should be calculated.

It might be:

- product list
- product
- basket
- checkout
- ...

The ContextId is used to configure, which set of Price Providers shall be used for calculating

prices.

CMS

CMS (content management system) is a software for the collective creation, editing and organi zation of content mostly in web pages, but also in other forms of media. This can consist text and multimedia documents. An author with access rights can such a system, in many cases wit h little programming or HTML knowledge use, since the majority of systems have a graphical user interface.

Source: Wikipedia

ERP

ERP (Enterprise resource planning) is business management software—typically a suite of integrated applications—that a company can use to collect, store, manage and interpret data from many business activities, including:

- Product planning, cost
- Manufacturing or service delivery
- Marketing and sales
- Inventory management
- Shipping and payment

Source: Wikipedia

Known ERP software packages:

- Microsoft Dynamics NAV (former navision)
- Microsoft Dynamics AX
- SAP

Scaled Prices

Scaled Prices is an array of configured prices that have additional information.

Those parameters have to match, if this scaled price is going to be shown on the website.

E.g. lets assume that in the backend there are scaled prices set like on the image below:

In the first case the price "**12**" including VAT will be shown on the website, if additional conditions will match:

- minimum quantity is 2
- the current date is between 2015-06-01 and 2015-07-01 22:00

Template

Usually a Twig template.

Template Debitor

Template debitor is a customer having a customer number which is defined in the ERP system. It is used to calculate e.g. prices for a special customer group when the customer does not have a customer number yet.

Usually theses settings are connected to a template debitor.

- prices setup
- VAT settings
- delivery costs

For details see StandardTemplateDebitorService

UBL Format

UBL (Universal Business Language) is a standard language used by eZ Commerce for the communication between the ERP systems and the shop. UBL is used also for representing

data in the shop such as the customer data.

VatCode

VatCode is a code (e.g. 'download', 'food'...) which defines which kind of VAT (%) applies to a product. The VatCode is stored in a ProductNode.

The Code usually is provided by the ERP-system or PIM. eZ Commerce provides a configuration which maps the Code to the corresponding VAT in percent.

See VatServiceInterface

Before you start

Please keep in mind that the Price Service is really connected with a lot of different modules in our shop. Be sure to check these out:

- Catalog Element
- ERP
- CustomerProfileData

Cookbook

The CookBook describes

- how to use the Price Engine in PHP
- how to implement an own Price Provider
- how to configure the Price Engine for situations in the shop
- how to pass additional information to priceRequest from catalogElement

API

See Price Engine - API

Templating

This section describes templates and twig functions which can be used for displaying and formatting prices and stock information.

FAQ

Does the price engine support volume based prices?

It depends on the Priceprovider.

The Priceprovider ERP will provide volume based prices as defined in the ERP.

How do i get delivery costs or additional costs?

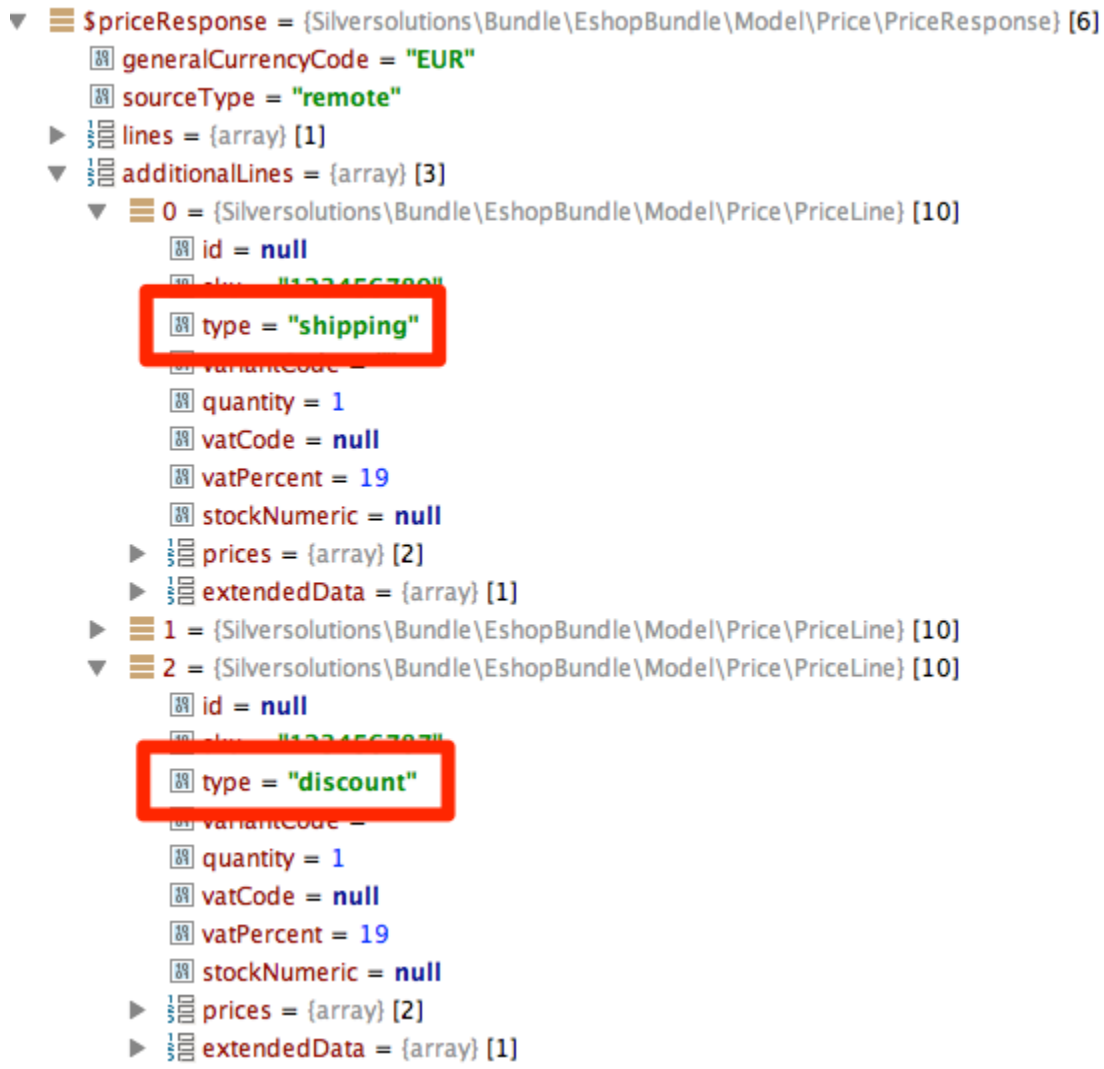
The price provider returns such a informations as *additional lines* with a special *type*.

```
$additionalLines = array();
$additionalLines[] = $this->createShippingPriceLine();
$priceResponse?setAdditionalLines($additionalLines);


/**
```

```
     * creates a PriceLine for shipping costs
     *
     * @return PriceLine
     */
    protected function createShippingPriceLine()
    {
        $priceLine = new PriceLine();

$priceLine->setType(PriceConstants::PRICE_RESPONSE_LINE_TYPE_SHIPPING
);
        $shippingCost =
$this->shippingCostCalculator->calculateShipping();
        $shippingVatCountry =
$this->configResolver->getParameter('shipping_vat_country',
'siso_core');
        $shippingVatCode =
$this->configResolver->getParameter('shipping_vat_code', 'siso_core');
        $vatPercent =
$this->vatService->getVatPercent($shippingVatCountry,
$shippingVatCode);

        //create PriceLineAmounts
        $shippingPrice = new PriceLineAmounts();
        $shippingCostVat = ($shippingCost * $vatPercent) / 100;
        $shippingCostNet = $shippingCost - $shippingCostVat;
        $shippingPrice->setLineAmountGross($shippingCost);
        $shippingPrice->setLineAmountNet($shippingCostNet);
        $shippingPrice->setLineAmountVat($shippingCostVat);
        $shippingPrice->setUnitPriceGross($shippingCost);
        $shippingPrice->setUnitPriceNet($shippingCostNet);
        $shippingPrice->setUnitPriceVat($shippingCostVat);
        // ToDo: Should we create a new price constant?

$shippingPrice->setSource(PriceConstants::PRICE_ENGINE_SOURCE_LOCAL);
        $prices[PriceConstants::PRICE_RESPONSE_PRICE_TYPE_CUSTOM] =
$shippingPrice;
        $prices[PriceConstants::PRICE_RESPONSE_PRICE_TYPE_LIST] =
$shippingPrice;
        $priceLine->setPrices($prices);
        // ToDO: check if we need to get the VAT for shipping costs.
        $priceLine->setVatPercent($vatPercent);
        $extendedData = array(
            'LineType' => 1,
            'CostType' =>
PriceConstants::PRICE_RESPONSE_LINE_TYPE_SHIPPING,
            'StockNumeric' => '',
            'AvailabilityColor' => '',
            'VatPercent' => $vatPercent,
            'PriceAmountGross' => $shippingCost,
            'PriceIsIncVat' => 1,
            'BelongsToLine' => '',
            'name' => 'msg.shipping_cost_name'
        );
```

```
            $priceLine->setExtendedData($extendedData);
```

```
        return $priceLine;
    }
```



How can i access stock information?

You can get stock information from the PriceLine in the PriceResponse

```
$priceResponse = $priceService->getPrices($priceRequest, $contextId);

foreach ($priceResponse->getLines() as $priceLine) {
 //get stock
 $stockNumeric = $priceLine->getStockNumeric();
 if (isset($stockNumeric)) {
     $stock = new StockField(array('stockNumeric' => $stockNumeric));
     $productNode->setStock($stock);
 }
}
```

How the currency is handled?

It depends on the Priceprovider.

- LocalPriceProvider is using the customer currency that is set in the PriceRequest.
- RemotePriceProvider is using the currency returned from the ERP and if not set, it is also using the customer currency.

How can i pass additional information to the price provider?

If your price provider needs some additional information, you can provide them in several ways:

- On the top level in *extendedData*

```
$priceRequest = new PriceRequest();
$extendedData = array(
    'customerId' => 126,
    'email' => 'test@testaccount.com'
);

$priceRequest->setExtendedData($extendedData);
```

- On the line level in *extendedData*

```
$priceLine = new PriceLine();
$extendedData = array(
   'remark'  => $customerRemark
);

$priceLine->setExtendedData($extendedData);
```

- You can also pass additional data in the parties, if they are connected to a customer

```
$priceRequest = new PriceRequest();
$buyerParty =
$customerProfileDataService->getDefaultBuyerParty();
$buyerParty->SesExtension->value['customerGroup'] = 'WIKI';

$priceRequest->setBuyerParty($buyerParty);
```

How can i find out which provider did calculate my prices?

If the price provider calculated the prices, it will set a source type in the PriceResponse. This source type can be evaluated.

Possible source types:

`PriceConstants::PRICE_RESPONSE_SOURCE_TYPE_REMOTE = 'remote'`

`PriceConstants::PRICE_RESPONSE_SOURCE_TYPE_LOCAL = 'local'`

```
//if the prices were not calculated by a remote source, set an error
message in the basket
if ($priceResponse->getSourceType() !==
PriceConstants::PRICE_RESPONSE_SOURCE_TYPE_REMOTE) {
    $basket->setErrorMessage(
        $this->transService->translate('Remote prices can not be
calculated!')
    );
}
```

What happens in case the ERP is not available?

For the basket the LocalPriceProvider will calculate prices as a fallback. The customer will see an error message that the realtime prices are not available at the moment - if the remote price provider was configured in the chain on the first place. See: When an error message is shown?

When an error message is shown?

When the chain for your context id was configured in a way, that the first provider is remote and the remote price calculation fails, en error message might be displayed. The shop will check the chain for your context id. So it is possible that in the basket there is an error message displayed if the remote price calculation failed, but not in the wishlit.

> You have to configure the service id of the remote price provider!

```
#configure the service id of the remote price provider
siso_price.default.price_service_chain_remote:
siso_price.price_provider.remote

siso_price.default.price_service_chain.basket:
 - siso_price.price_provider.remote
    - siso_price.price_provider.local

siso_price.default.price_service_chain.stored_basket:
 - siso_price.price_provider.local

siso_price.default.price_service_chain.wish_list:
 - siso_price.price_provider.local
```

Why can user see/not see prices on product detail page with/without the necessary role?

Caching has to be configured to use user-hash in order to display the correct product detail page to anonymous or registered users. If this is not done, the first call will be cached for all users!

Why can user not see prices in sliders, catalog list, product detail or comparison?

They don't have the necessary role to see prices, it has to be configured for users in the backend. (siso_policy/see_product_price)