Advanced Product Feeds 1.0.41

How to install extension

1. Backup your store database and web directory.
2. Login to SSH console of your server and navigate to root directory of Magento 2 store.
3. Copy installation instructions from page My Downloadable Products to SSH console and press ENTER.
4. Run command php -f bin/magento module:enable Mirasvit_Core Mirasvit_Feed Mirasvit_Report for enable extension.
5. Run command php -f bin/magento setup:upgrade to install extension.
6. Run command php -f bin/magento cache:clean for clean cache.
7. Deploy static view files

   rm -rf pub/static/*; rm -rf var/view_preprocessed/*; php -f bin/magento setup:static-content:deploy

How to upgrade extension

To upgrade extension follow next steps:

1. Backup your store database and web directory.
2. Login to SSH console of your server and navigate to root directory of Magento 2 store.
3. Run command composer update mirasvit/module-feed for update extension sources.
4. Run command php -f bin/magento module:enable Mirasvit_Core Mirasvit_Feed Mirasvit_Report for re-enable extension.
5. Run command php -f bin/magento setup:upgrade for install updates.
6. Run command php -f bin/magento cache:clean for clean cache.
7. Deploy static view files

rm -rf pub/static/*; rm -rf var/view_preprocessed/*; php -f bin/magento setup:static-content:deploy

Disabling Extension

Temporary Disabling

To temporary disable extension please follow the next steps:

1. Login to SSH console of your server and navigate to root directory of Magento 2 store.
2. Run command php -f bin/magento module:disable Mirasvit_Feed for disabled extension.
3. Login in to Magento back-end and refresh store cache (if enabled).

Extension Removing

To uninstall extension please follow the next steps:

1. Login to SSH console of your server and navigate to root directory of Magento 2 store.
2. Run command composer remove mirasvit/module-feed for remove extension.
3. Login in to Magento back-end and refresh store cache (if enabled).

List of the pre-installed templates

Template name	Format
Amazon(inventory)	XML
Amazon(Marketplace)	XML
Amazon(pictures)	XML
Amazon(price)	XML
AmazonAds	TXT
Become Europe	CSV
Beslist	XML
Billiger.de	CSV
Bing Shopping	TXT
CJ	XML
eBay(Commerce Network)	CSV
eBay.com (Store)	CSV
Facebook (storefront)	CSV

Template name	Format
Fishpond	CSV
GetPrice Categories	XML
GetPrice Products	XML
Google Shopping	XML
Google Shopping (configurable products)	XML
Google Shopping Review	XML
Google Shopping Update	XML
idealo.it	CSV
it.bestshopping.com	CSV
Kelkoo	XML
Kieskeurig	XML
LeGuide.com	TXT
Newegg	XML
Newegg(inventory)	XML
Nextag	TXT
pagineprezzi	TXT
Partner-Ads	XML
PriceGrabber	TXT
PriceMe	XML
PriceRunner	XML
PriceSpy	TXT
Rakuten (Apparel)	TXT
Sears.com Inventory	XML
Sears.com Item	XML
Sears.com Price	XML
ShareASale	CSV
ShopMania	XML
Shopping.com	XML
ShopPrice	XML
Shopzilla	TXT
SingleFeed	CSV
The Find	CSV
TradeDoubler	CSV
TradeTracker	CSV
Twenga	CSV
Webgains	CSV
Yandex Market	XML

Manage templates

To manage templates go to **Products Advanced Product Feeds Templates**.

Extension includes more then 45 ready to use templates for all popular price comparison engines (Google Shopping, Amazon, eBay, Shopzilla, etc.).

Note

Before you create a new template, you need to check product feed specification for the specified marketplace (template format type, fields delimiter, required fields, etc).

You can create template for any comparison shopping engine.

To create a new template, follow these steps:

1. Go to **Products Advanced Product Feeds Templates**, then click the **Add Template** button in the upper righthand corner.
2. Fill in the following fields:
    - **Name** - name of the new template.
    - **File Type** - feed output format. There are 3 types are available for data feed:
        - **CSV** - a comma-separated values, each item placed on a new line. File extension is *.csv*.
        - **TXT** - same as CSV file, but with *.txt* extension.
        - **XML** - uses tags to define blocks of content. Information about your items is enclosed within these tags, which are indicated by angle brackets. File extension is *.xml*.
    - At **Content Settings** tab, you you need configure template depend requirements and file type:
        - CSV, TXT
        - XML
3. Click the **Save** button

How to create a new data feed

Note

Pre installed templates may require additional attributes and settings by reason of the specific selling items or country location. Check products attribute requirements using marketplace specifications.

To create a new data feed, follow these steps:

1. Go to **Product Advanced Product Feeds Feeds**. Press button **Add Feed**.
2. Select one of the existing template to create a feed. To create an empty feed, select **Empty Template**.
3. Press button **Continue**.
4. Fill few requirement fields:
    - **Name** - name of the data feed.
    - **Filename** - name of the data feed file. File will be located at [magento_path]/media/feed/filename.
    - **Store View** - store view, for which will be generated data feed.
    - **Is Active**

    Additionally, if you selected **Empty Template**, you need fill these fields:

    - **File Type** - there are three file types available for data feed.
        - **CSV** - a comma-separated values, each item placed on a new line. File extension is *.csv*.
        - **TXT** - same as CSV file, but with *.txt* extension.
        - **XML** - uses tags to define blocks of content. Information about your items is enclosed within these tags, which are indicated by angle brackets. File extension is *.xml*.
5. Press button **Save and Continue Edit**.
6. To generate data feed, press button **Generate** at the top right corner.

FTP settings

Extension can automatically deliver data feed file via FTP to Shopping Engine Service.

Note

Check marketplace merchant account for FTP details, or ask about FTP credentials at marketplace Support Center

To configure FTP delivery, follow these steps:

1. Open tab **FTP Settings** at feed edit page.
2. At tab you need to enable FTP delivery and fill these fields:
    - **Protocol** - you can select FTP or SFTP connection.
    - **Host Name** - this is a FTP server where you would like to send your feed.
    - **User Name** - the username to FTP server.
    - **Password** - the password to FTP server.
    - **Path** - optional field, enter path to your merchant folder provided by Shopping Engine Service.
    - **Passive mode** - most FTP servers work in Passive mode, even when you use a firewall.
3. Press button **Save And Continue Edit**.

After enabling FTP delivery, you can run delivery of feed manually by pressing button **Delivery Feed** at the right top corner.

Additionally extension can deliver feed by schedule, after each feed generation.

Schedule Task Settings

You can configure your feed to automatically generate data feed file by schedule.
If FTP settings are enabled, feed will be automatically delivered to the marketplace after generation by schedule.

Note

To generate feed by schedule, magento cron must be configured. See How to Setup Cron for Magento.

To configure schedule follow these steps:

1. Open tab **Scheduled Task** at feed edit page.
2. Set **Status** - **Enabled** to enable feed generating by schedule
3. Set up the following fields:
    - **Day** - days of the feed generation.
    - **Time** - time of the feed generation.
4. Press button **Save And Continue Edit**.

For example, if selected **days** are *Monday*, *Wednesday* and **time** *03:00AM*, *05:00AM*, then feed will be generated 4 times during a week.

~! If you have a few feeds with scheduled tasks, to prevent cron job errors you need to set different scheduler **time** for each feed.

Email Notifications

Open your feed **Additional** tab at feed edit page.

Extension can automatically send email notifications for next events:

- Successful Generation
- Unsuccessful Generation
- Successful Delivery
- Unsuccessful Delivery

Fill in the following lines:

- **Email Addresses** - email addreses for email notifications (use comma separator to set more then one email address)
- **Emails** - sets events for further email notification

Press the button **Save And Continue Edit**.

Google Analytics Campaign

Note

Google Analytics should be configured and activated in order to use this feature.

Extension can automatically append google analytics campaign parameters to your product URLs.

To configure **Google Analytics Campaign**, follow these steps:

1. Open tab **Google Analytics** at feed edit page.
2. Fill in 3 required fields:
    - **Campaign Source** - Identifies a search engine, newsletter name, or other source.(i.e.

google, citysearch, newsletter)
    - **Campaign Medium** - Identifies a medium such as email or cost-per-click. (i.e. cpc, banner, email)
    - **Campaign Name** - Identifies a specific product promotion or strategic campaign. (i.e product, promo code, or slogan)
      Also, you can optionally fill in other fields:
    - **Campaign Term** - Identifies paid keywords.
    - **Campaign Content** - Differentiates ads or links that point to the same URL.
3. Press button **Save And Continue Edit**.

After adding google analytics parameters, you need to generate your feed. In feed file all product URLs will be http://example.com/product.html?fep=…&fee=…&utm_source=…&utm_medium=…&utm_name=…
After feed generating, you don't need to do additional configuration adjustments.
Additionaly, in campaign fields, you can use any pattern.
To track **Google Analytics Campaign** log in into your account and go to **Traffic Sources** > **Campaigns**. Select campaign source from the list.

Additional Settings

Extension allows to set up additional settings for feed export.
Go to **Catalog > Manage Feeds** and open your feed. Open tab **Additional**.

Export Configuration

- **Export Only Enabled Products** - if option is enabled, products with status **Disabled** will be excluded from the feed export.
- **Export Only New And Changed Products** - if option is enabled, only new and changed products will be exported since the last time feed generation.
  You can reset already exported products, by pressing on link **Reset Exported Products**.

Note

This feature will work, only if number of assigned to feed filters greater than zero.

- **Enable archiving** - if option is enabled (**.zip**), after feed generation extension also generate *zip* archiv of the same feed.

Reports Configuration

For each feed, you can enable/disable tracking clicks and orders by changing setting **Enable Reports**.

If feature enabled, extension append to product url two special arguments (fee=, fep=) to track clicks and orders (http://example.com/product.html?fee=1&fep=3), where **fee** - ID of the feed, and **fep** - ID of the product).

Content Filter

This option alllows to remove symbols from the feed content.

To make option enabled, fill in the following fields:

- **Allowed Characters** - Sets allowed characters for the product feed content. Leave empty to allow all characters. Begin and end with the "/" to use as a regular expression (ex. /[A-Za-z]*/).
  Begin from any other character to accept only particular characters (ex. ABCDEFGH).
- **Ignored Characters** - Sets ignored characters for the product feed content. Leave empty to accept all characters. Begin and end with the "/" to use as a regular expression (ex. /[A-Za-z]*/).
  Begin from any other character to ignore only particular characters (ex. ABCDEFGH).

How to configure XML Feed

If you select **XML** file type at tab **Content Settings** you can create/edit xml schema for your feed.

By default we provide templates for XML feeds, so you can easily copy it and change for your requirements.

Usually Comparison Shopping Engines provide a template of the xml file. Based on this template, you can create own xml schema.

Typical xml schema:

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0" xmlns:g="http://base.google.com/ns/1.0">
   {% for product in context.products %}
    <item>
      <attribute_1><![CDATA[{{ product.attribute_1 }}]]></attribute_1>
      <attribute_2><![CDATA[{{ product.attribute_2 }}]]></attribute_2>
      .....
    </item>
   {% endfor %}
```

</rss>

Product cycle block:

{% for product in context.products %}
...{% endfor %}

Inside this block, you can use any product attribute.

Category cycle block:

{% for category in context.categories %}
...{% endfor %}

Inside this block, you can use any category attribute.

Review cycle block:

{% for review in context.reviews %}
...{% endfor %}

Inside this block, you can use any review attribute.

Attribute (pattern) block:

<attribute_1><![CDATA[{{ product.attribute_1 }}]]></attribute_1>
<attribute_1><![CDATA[{{ category.name }}]]></attribute_1>
<attribute_1><![CDATA[{{ review.nickname }}]]></attribute_1>

The attribute code must be enclosed in double curly brackets: *{{ product.attribute_code }}*. You can use all attribute codes available at **Store > Attributes > Product** and all static attributes (ex. entity_id, created_at etc).

Additionally in curly brackets you can place any available pattern. [Full pattern list of patterns]

Note

Characters like < and & are illegal in XML elements.

- < will generate an error because the parser interprets it as the start of a new element.
- & will generate an error because the parser interprets it as the start of an character entity.

We suggest enclosed all patterns in **CDATA** block <attribute><![CDATA[{{ pattern }}]]></attribute>. In this case, xml data feed will be valid.

How to configure CSV, TXT Feed

If you select **CSV** or **TXT** file type at tab **Content Settings**, you can create/edit attribute schema for your feed.

Note

When you use pre-installed templates, you need to check if the attributes from the template response for the same values as your store attributes. If this attribute doesn't exist in your store, set appropriate product attribute or pattern for the same line.

Content Settings

Before creating attribute scheme, you need to fill in required file settings:

- **Fields Delimiter** - delimiter, which allows you to split text into columns in your feed file. Supported delimiters are:
  - Comma ","
  - Tab "\t"
  - Colon ":"
  - Space " "
  - Vertical pipe "|"
  - Semi-colon ";"
- **Fields enclosure** - allows enclose data in your feed file.
- **Include Header** - set "Yes" to include a header row (attribute names) in the first line of your feed file.
- **Extra header** - set "Yes" to include an additional header row in the first line of your feed file. It will always be above the first attributes row or the columns header.

Field Mapping

In field mapping table you can add/remove rows, change rows ordering, set output type, symbols limit. Each row in mapping table is a column in data feed file.

For adding new column to your CSV feed, you need to create a new row and fill it with few params:

- **Field Name** - the header column name.
- **Prefix** - allows you place prefix before each value in column.

  Example

  The prefix *Special* for attribute *Name*, will return values *Special Product Name 1*, *Special Product Name 2*.
- **Type** - following types available:
  - **Attribute** - allows to select any store attribute from the drop down list.
  - **Parent Attribute** - allows to export configurable products. In this case, simple associated products will have attribute values of the parent attribute.
    If you have configurable products, we suggest to use this Type with fields: "Product URL", "Grouped id"
  - **Pattern** - this option allows you to enter static value or use patterns
- **Value** - you need to select the attributesribute or put pattern
- **Suffix** - allows you place suffix after each value in a column.
- **Output Type** - following types available:
  - **Default** - value not will be changed
  - **Integer** - value will be converted to ceil number
  - **Price** - value will be converted to price format (#.##)
  - **Strip Tags** - extension will clear text from html tags
- **Symbols Limit** - limit on number of symbols in value

List of Patterns

All patterns must be enclosed in curly brackets. In patterns you can use codes of attributes, filters, links to parent products, base php functions and calculations.

The base pattern schema {{ entity.attribute | filter | filter }}

Attribute Patterns

-

{{ product.entity_id }} - ID of the product

- {{ product.sku }} - an identifier of the product
- {{ product.name }} - a name of the product
- {{ product.description }} - a description of the product
- {{ product.short_description }} - a short description of the product
- {{ product.status }} - a status of the product

  Possible values:

  - Enabled
  - Disabled
- {{ product.visibility }} - a visibility of the product

  Possible values:

  - Not Visible Individually
  - Catalog
  - Search
  - Catalog, Search
- {{ product.url_key }} - a url key of the product
- {{ product.url }} - a direct url to the product
- {{ product.price }} - price of product (without discounts, catalog rules etc)
- {{ product.regular_price }} - a regular price of the product
-

{{ product.final_price }} - a final price (saleable) of the product

The price of product after applying special price and catalog price rules.

- {{ product.special_price }} - a special price of the product

  The special price of the product.

  Special price ignore values of *Special Price From Date* and *Special Price To Date*

- {{ product.regular_price }} - a regular/base price of the product
- {{ product.tax_rate }} - a tax rate for the product
- {{ product.category }} - a name of the assigned category to the product

Note

If product is assigned to a few categories, extension selects **Category** using next logic:
There always is selected the most nested category. For example, if a product is assigned to a few categories at different level, the attribute {category} returns the **name** of the category that is the most nested in the category tree.
If product is assigned to a few categories at the same level, the extension selects a category with the lowest position of the product. Change position of the product you can at **Catalog > Manage Categories**, tab **Category Products**

- {{ product.category.id }} - a ID of the assigned category to the product

Note

If product is assigned to a few categories, extension selects **Category Id** using next logic:
There always is selected the most nested category id. For example, if a product is assigned to a few categories at different level, the attribute {category_id} returns the **id** of the category that is the most nested in the category tree.
If product is assigned to a few categories at the same level, the extension selects a category id with the lowest position of the product. Change position of the product you can at **Catalog > Manage Categories**, tab **Category Products**

- {{ product.category.path }} - a path of the category names

  E.g. Computers > Notebooks > Apple

Note

If product is assigned to a few categories, extension selects **Category Path** using next logic:
There always is selected the most nested category path. For example, if a product is assigned to a few categories at different level, the attribute returns the **path** that is the most nested in the category tree.
If product is assigned to a few categories at the same level, the extension selects a category with the lowest position of the product. Change position of the product you can at **Catalog > Manage Categories**, tab **Category Products**

- {{ product.category.url }} - a direct url of the assigned category to the product

  The direct url to parent category.

Note

If product is assigned to a few categories, extension selects **Category Url** using next logic:
There always is selected the most nested category url. For example, if a product is assigned to a few categories at different level, the attribute {{ product.category.url }} returns the **url** of the category that is the most nested in the category tree.
If product is assigned to a few categories at the same level, the extension selects a category id with the lowest position of the product. Change position of the product you can at **Catalog > Manage Categories**, tab **Category Products**

- {{ product.attribute_set }} - a name of the assigned attribute set to the product
-

{{ qty }} - a quantity of the product

- {{ product.is_in_stock }} - a stock status of the product

  Possible values:

  - **0** - Out of Stock
  - **1** - In Stock
- {{ product.image }} - a direct url to base image of the product
- {{ product.thumbnail }} - a direct url to thumbnail image of the product
- {{ product.small_image }} - a direct url to small image of the product
- {{ product.gallery[0] }}, {{ product.gallery[1] }} ... - a direct url to gallery images of the product
- {{ product.rating_summary }} - average product rating (from 0 to 5)
- {{ product.reviews_count }} - number of approved reviews

Parent product values

You can use suffix **.parent** ({{ product.parent.name }}, {{ product.parent.price }}, {{ product.parent.url }} etc), if you need return value of parent product.

Example

If current product associated with configurable/grouped/bundled product, pattern {{ product.parent.url }} , will return URL to parent product. If extension can't find parent product, it uses current product.

Note: Parent suffix is very useful when you export **simple** products with visibility **Not Visible Individually**. In this case, product can't have a direct link, so you must use a link to the parent product.

Examples

```
{% for product in context.products %}
   {% for image in product.gallery %}
   <picture>{{ image }}</picture>
   {% endfor %}
   <created>{{ product.created_at | dateFormat: 'd.m.Y H:i:s' }}</created>{% endfor %}
```

Filters

Filters are simple methods that modify the output of numbers, strings, variables and arrays. They are placed within an output tag {{ }} and are separated with a pipe character |.

String/HTML Filters

- lowercase - converts a string into lowercase.

  {{ product.name | lowercase }}

  Original: Dash Digital WatchOutput: dash digital watch

- uppercase - converts a string into uppercase.

  {{ product.name | lowercase }}

  Original: Dash Digital WatchOutput: DASH DIGITAL WATCH

- replace - replaces all occurrences of a string with a substring.

  {{ product.name | replace: 'Digital', 'Analog' }}

  Original: Dash Digital WatchOutput: Dash Analog watch

-

append - appends characters to a string.

{{ product.name | append: ' - best choice' }}

Original: Dash Digital WatchOutput: Dash Digital Watch - best choice

- prepend - prepends characters to a string.

  {{ product.name | prepend: 'Best choice - ' }}

  Original: Dash Digital WatchOutput: Best choice - Dash Digital Watch

- capitalize - capitalizes the first word in a string.

  {{ product.color | capitalize }}

  Original: dark redOutput: Dark red

- escape - escapes html tags in a string.

  {{ product.description | escape }

- nl2br - inserts a
 linebreak HTML tag in front of each line break in a string.

  {{ product.short_description | nl2br }

- remove - removes all occurrences of a substring from a string.

  {{ product.name | remove: 'Digital' }}

  Original: Dash Digital WatchOutput: Dash Watch

- stripHtml - strips all HTML tags from a string.

{{ product.description | stripHtml }}

- truncate - truncates a string down to 'x' characters.

  {{ product.name | truncate: '15' }}

  Original: Dash Digital WatchOutput: ash Digital Wa

- plain - converts any text to plain format.

  {{ product.description | plain }}

- ifEmpty - return argument, if value is empty string

  {{ product.color | ifEmpty: 'Multi color' }}

  Original: Output: Multi color

- dateFormat - converts string to specified date-time format.

  {{ product.created_at | dateFormat: 'd.m.Y H:i' }}

  Original: 2016-02-18 10:11:12Output: 18.02.2016 10:11

- json - converts object or array to JSON format.

  {{ product.gallery | json }}

Number Filters

- ceil - rounds an output up to the nearest integer.

{{ product.weight | ceil }}

Original: 1.423 Output: 2

- floor - rounds an output down to the nearest integer.

  {{ product.weight | floor }}

  Original: 1.423 Output: 1

- round - rounds the output to the nearest integer or specified number of decimals.

  {{ product.weight | round: '2' }}

  Original: 1.423 Output: 1.42

  {{ product.weight | round }}

  Original: 1.423 Output: 1

- numberFormat - formats number to specified format (php function).

  {{ product.price | numberFormat: '2', '.', ',' }}

Price/Currency Filters

- price - formats price to default format.

  {{ product.price | price }}

  Original: 100.42 Output: 100.42

-

convert - converts price from base currency to specified currency.

{{ product.price | convert: 'EUR' }}

Original: 100Output: 92.28

- inclTax - include tax to product price

  {{ product.price | inclTax | price }}

- exclTax - exclude tax from product price

  {{ product.price | exclTax | price }}

Array Filters

- first - return first element in array.
- last - return last element in array.
- join - join array to string using glue.
- count - return number elements in array.
- select - select values for key from array.

URL Filters

- secure - return secure url (with https://)
-

unsecure - return unsecure url (with http://)

- mediaSecure - return media url using "Base URL for User Media Files"

  {{ product.image | mediaSecure }}

  Original: http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg
  Output: https://cdn-secure.com/catalog/product/m/h/mh03-black_main.jpg

- mediaUnsecure - return media url using "Secure Base URL for User Media Files "

  {{ product.image | mediaUnsecure }}

  Original: http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg
  Output: http://cdn.com/catalog/product/m/h/mh03-black_main.jpg

Image Filters

- resize - resize image

  {{ product.image | resize: 100, 100 }}

  Original: http://example.com/pub/media/catalog/product/m/h/mh03-black_main.jpg
  Output: http://example.com/pub/media/cache/200x200/catalog/product/m/h/mh03-black_main.jpg

---

Note

- {{ product.name | truncate: '150' }}
- {{ product.description | plain | truncate: '1000' }}
- {{ product.weight | round: '2' }} kg
- {{ product.manufacturer | ifEmpty: 'not set' }}

Dynamic attributes

Dynamic attribute - it's attribute, that can return values depends on conditions or value of another attributes.

To create a new dynamic attribute, follow these steps:

1. Go to **Product Advanced Product Feeds Dynamic Attributes**. Press button **Add Attribute**.
2. Fill in the following fields:
    - Name - name of the dynamic attribute
    - Code - code of the dynamic attribute. You will see this code when you select this dynamic attribute in the templates.
    - Set conditions:
        - Press "Add "OR" condition"
        - Press "Add "AND" condition"
        - Select attribute at left corner
        - Select condition
        - Select or input condition value
        - Select Output Type
        - Select attribute or input pattern - when condition is true, dynamic attribute will return this value.
    - Press button Save.

    To set default value, just leave Conditions column without conditions.

Dynamic variables

Dynamic variable - its user defined php code, that mut return string value.

To create a new dynamic variable, follow these steps:

1. Go to **Product Advanced Product Feeds Dynamic Variables**. Press button **Add Variable**.
2. Fill in the following fields:
    - Name - name of the dynamic variable
    - Code - code of the dynamic variable. You will see this code when you select this dynamic

variable in the templates.
    - PHP Code

Example

- Name: GTIN
- Code: gtin

PHP Code:

```
$gtin = $product->getGtin();
if (!$gtin) {
   $gtin = $product->getId();
   $gtin .= str_repeat('0', 12 - strlen($gtin));
}return $gtin;
```

Usage:

```
{{ product.variable.gtin }}
```

Output:

```
124200000000
```

Command Line Interface

Usage: php -f bin/magento [options]

- mirasvit:feed:export - generate all active feeds
- mirasvit:feed:export --id=<id> - generate feed with specified id
- mirasvit:feed:delivery - delivery all active feeds
- mirasvit:feed:delivery --id=<id> - delivery feed with specified id