WHITE PAPER - TRIMIT PORTALS SCHEDULES

This document contains information about the Schedules functionality in TRIMIT Portals for Microsoft Dynamics Business Central BC13 R2

Version: BC13.2
Date: 24-01-2020

CONTENTS

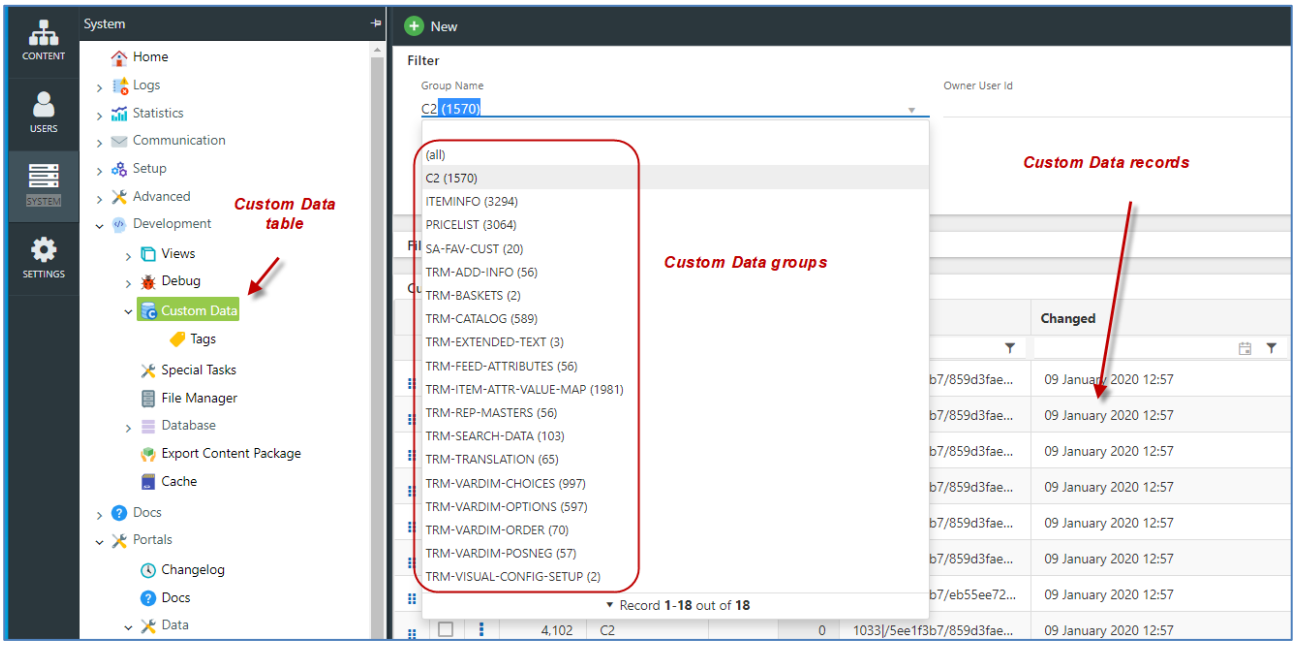
INTRODUCTION

Schedules functionality takes care of running a variety of jobs initiated in Modelyn without any user intervention. Primarily used for data synchronization between TRIMIT and Modelyn.

Possible schedule application areas:

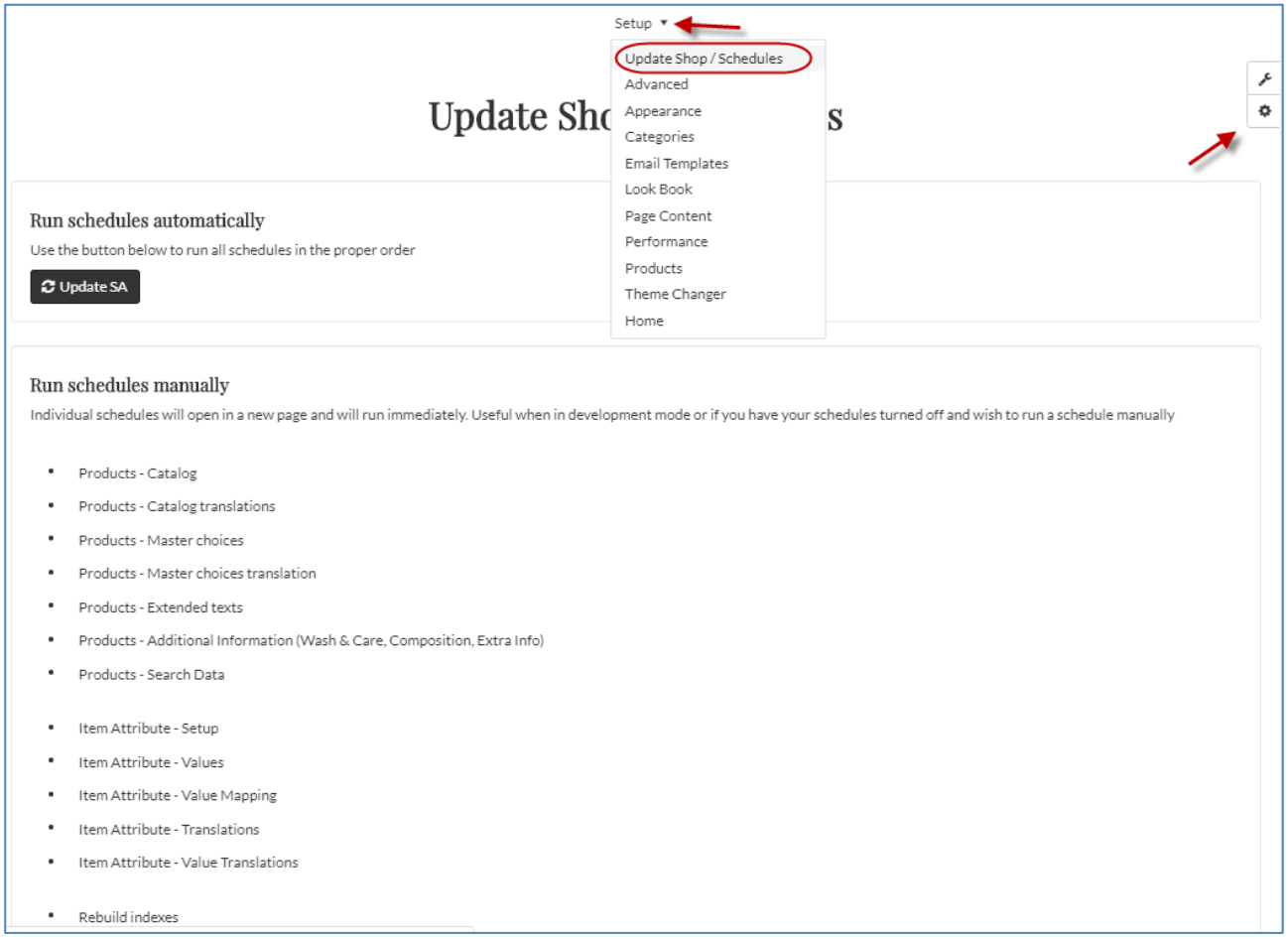
- *Synchronizations of data sets (i.e. Category Items) from TRIMIT to a Modelyn database.*
- *Populating a web server cache. Enables Portal users to experience much faster performance from the first moment they enter the Portal. The cache is always populated and up to date.*
- *And more…*

The portals data synchronization between TRIMIT and Modelyn is based on schedules which populate a ***Custom Data*** table in Modelyn database. This increases the web site performance comparing to direct request/response calls between the Modelyn and TRIMIT.
Custom Data table contains data which is taken from TRIMIT by means of Modelyn schedules. This data is then used on SA portal, B2C and B2B Webshops.
Custom Data includes, among other - product, availability and price information. Using Custom Data methodology together with schedules improves site performance, saving the round trip to TRIMIT for data retrieval.



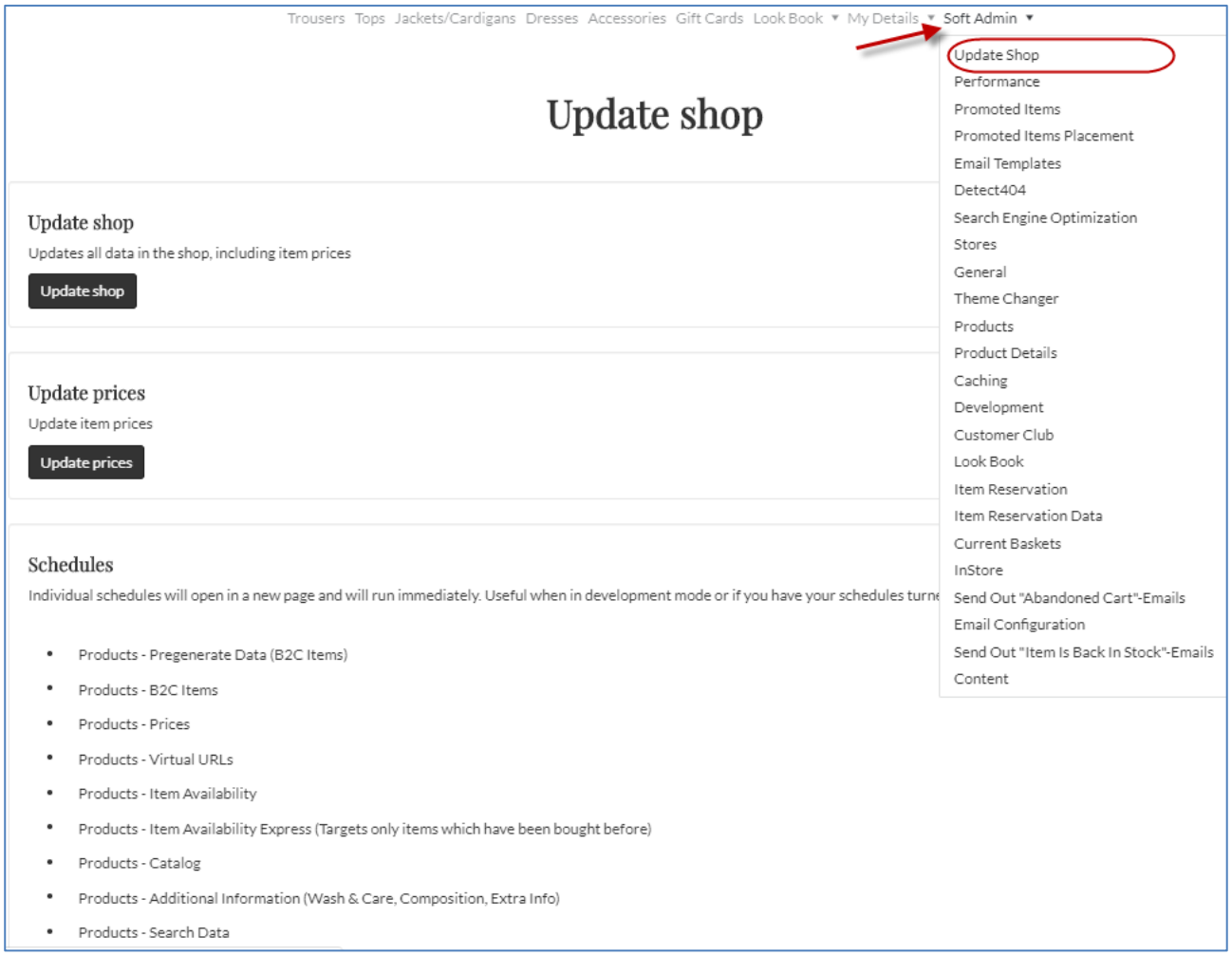
SCHEDULES

When enabled, schedules are executed automatically by the Modelyn engine.
It is also possible to start the main schedules from the portal's Soft Admin pages and from PIR dashboard or from master admin page (this synchronizes data for individual master only).
To manually run schedules from the soft admin pages on SA portal and B2B Webshop - login as admin user and navigate to soft admin (**Setup** link in the header of the page) and go to **Update Shop / Schedules.**



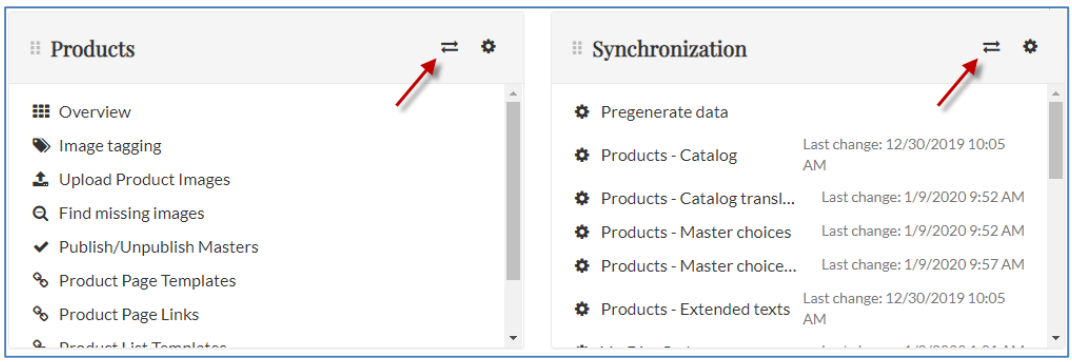
Here it is possible to initiate running all SA portal related schedules at ones by clicking **Update SA.**
It is also possible to initiate running individual SA portal related schedules under ***Run schedules manually*** area. Click on schedule to run it.
B2B Webshop has identical setup.

To manually run schedules from the soft admin pages on the B2C webshop - login as admin user and click **Soft Admin** menu (alternatively hover mouse over **Soft Admin** and in the drop down select ***Update shop***)

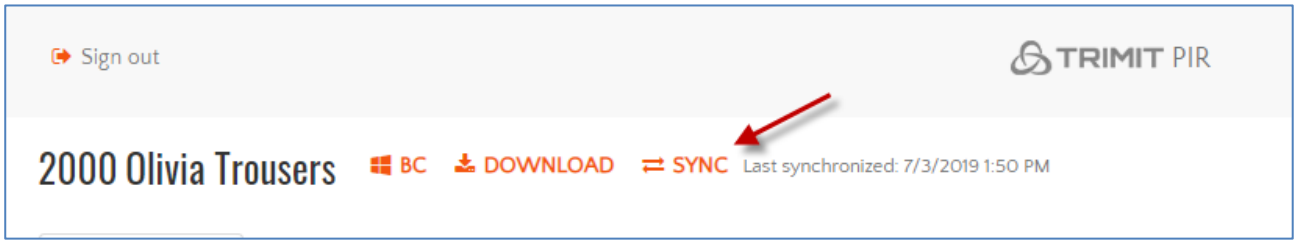


Here it is possible to initiate running all B2C Webshop related schedules at ones by clicking **Update shop.** Moreover, it is possible to synchronize only product prices by clicking **Update prices**.
To initiate running individual B2C Webshop related schedules click on specific schedules under ***Schedules*** area.

B2B and B2C webshops and SA portal schedules can also be initiated from the PIR dashboard. In this case the synchronization with TRIMIT BC will affect all three portals. This can be done on *Products* and *Synchronization* docks. The schedules will run in a background.



If data has to be synchronized only for a single master, then it is possible to run schedules from a master admin page in PIR. This page can be opened either by searching for master or via *Overview* in **Products** dock.



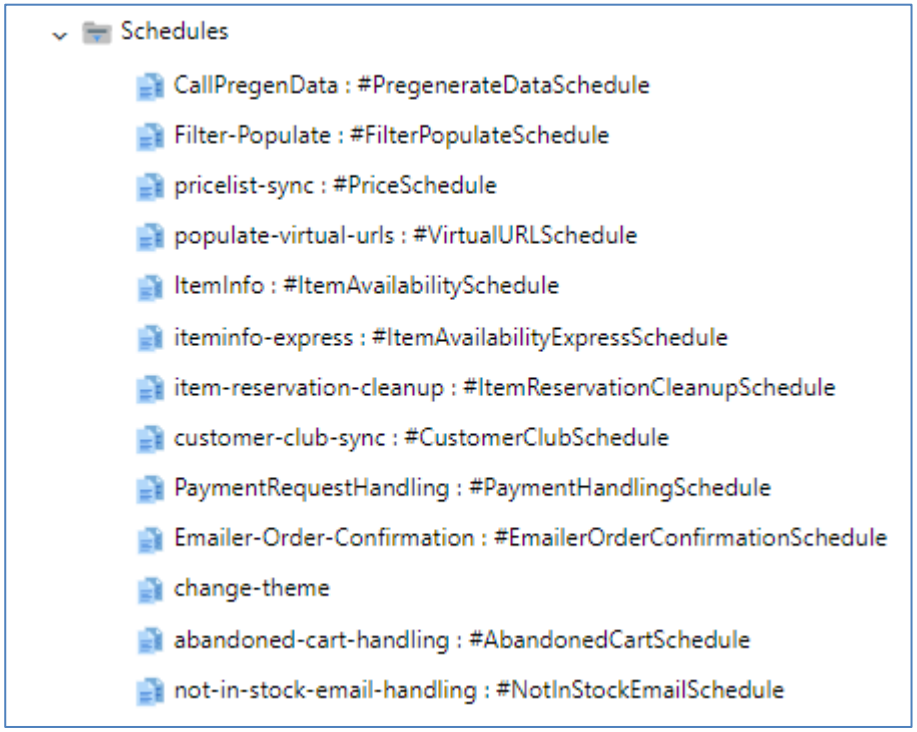
Running schedules for individual master also updates all three portals.

On portal reboot, the schedules start execution according to cron and simple expressions.

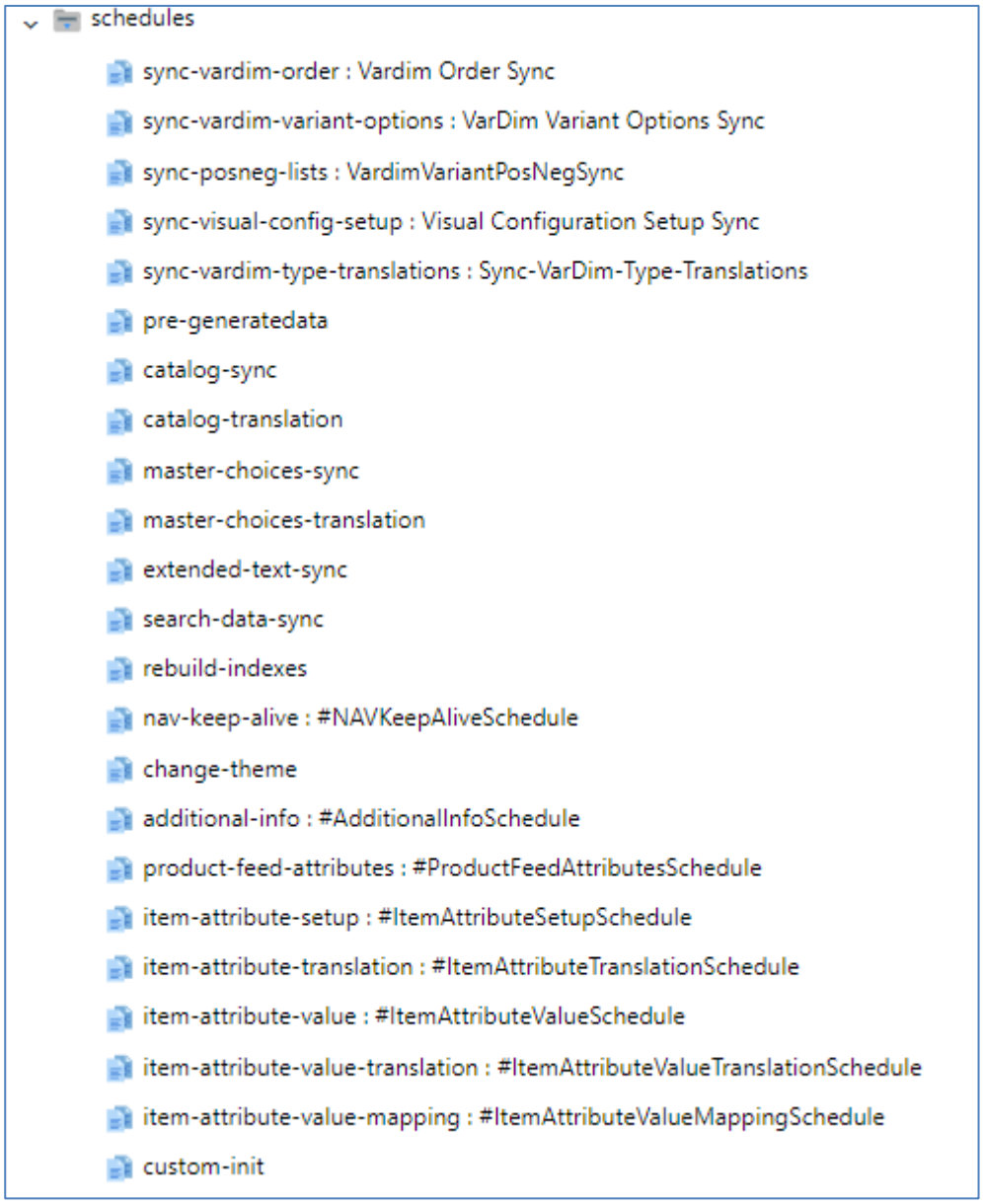
Schedules last run time stamp is only updated if there have been any changes. The update in schedules synchronization time stamp on master admin page reacts only on change in master choices.

SCHEDULES STRUCTURE IN MODELYN

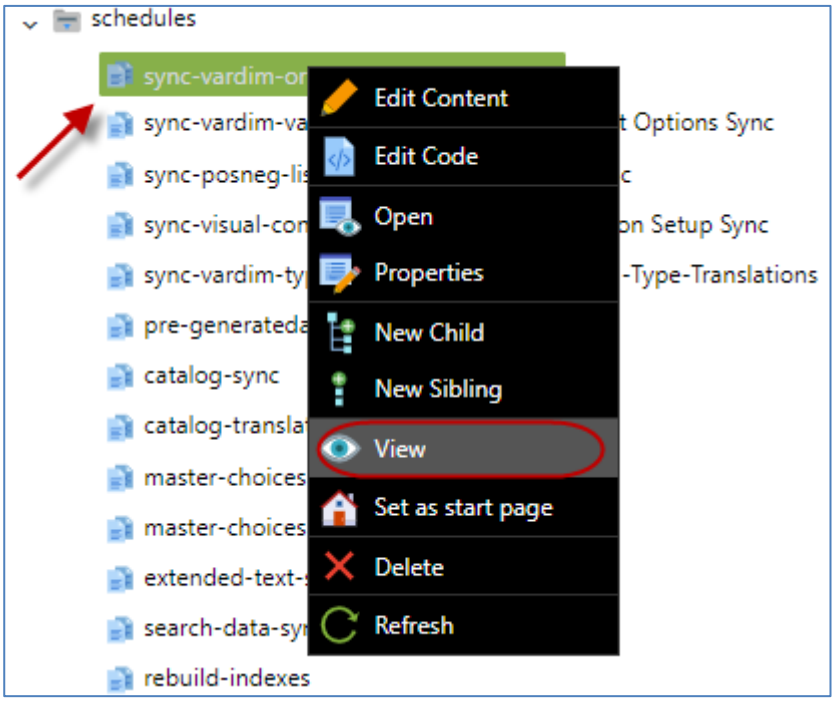
All the B2C related schedules reside in Modelyn under ***Content/Content Structure/B2C/SystemPages/Schedules***.



All the SA portal and B2B Webshop related schedules reside in Modelyn under ***Content/Content Structure/pir/schedules***. Here it is also possible to find schedules which are common for all portals including B2C (e.g. Attributes synchronization schedules).

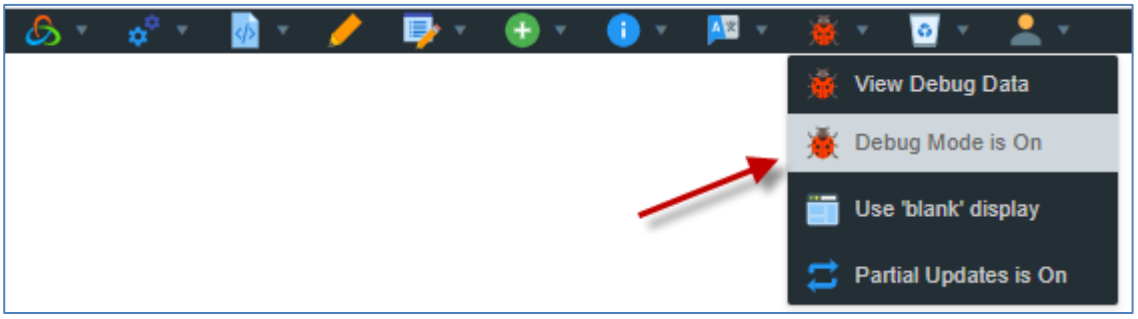


A schedule can be initiated from here by right-clicking on it and selecting ***View*** (this will open a new browser tab).

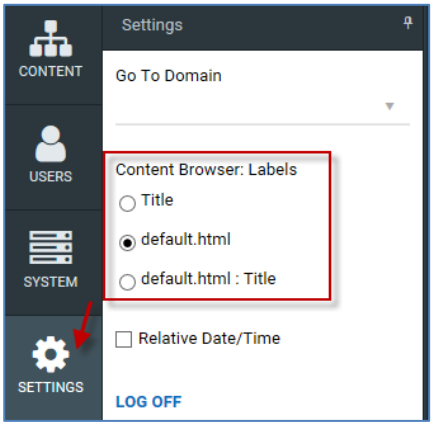


Be aware that running B2C schedules as described above should be done from B2C domain, e.g. *b2c.<my domain>.com/cpoadmin.* And running schedules under ***Content/Content Structure/pir/schedules*** should be done from PIR domain, e.g. *pir.<my domain>.com/cpoadmin.* Otherwise, the portal can respond with the following error: "***xhtml display for 'scheduledContent' not found in view 'Portals'***".

In case of permission warning – enable debugger mode in Modelyn



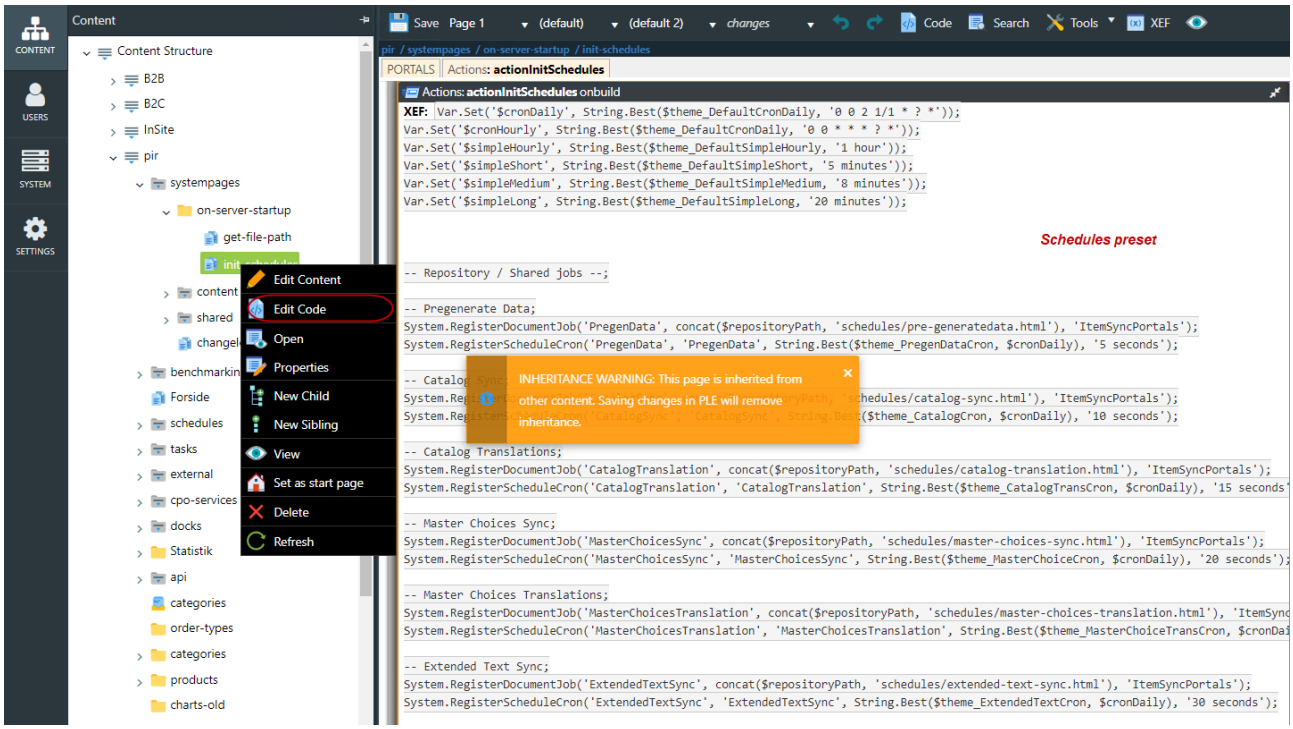
The schedules representation view can be changed as follows:



SETUP

Schedules registration in Modelyn is done after website reboot (or server restart or application pool restart). That is why it is important to reboot the application pool right after an installation of a new portal or after upgrade to a latest Modelyn version. Any modification in schedules also requires a website reboot.

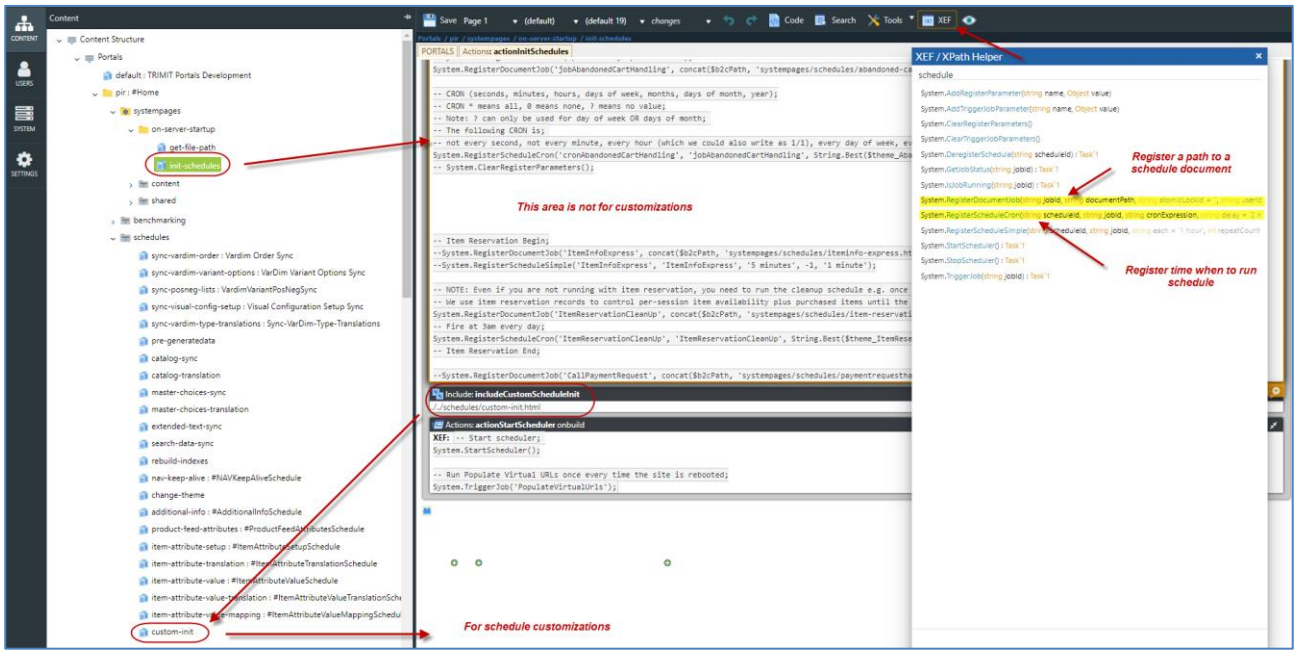
The schedules registration functions are preset under ***Content / Content Structure / pir / systempages / on-server-startup / init-schedules***.



This area in not for customizations and editing is not allowed here. Instead, schedule can be published, unpublished or modified in customization schedule document under ***Content / Content Structure / Portals / pir / schedules / custom-init***.

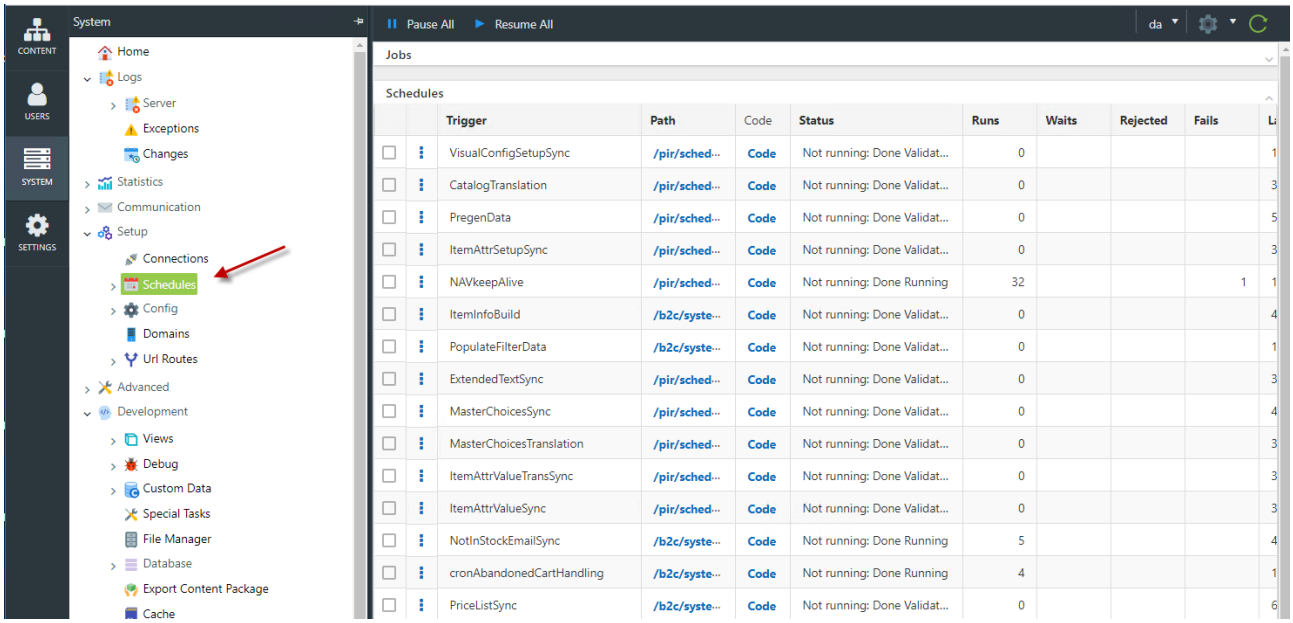
The way the schedules are executed is as follows:

- When the portal is rebooted, then the default schedules which are specified under action: **actionInitSchedules** are being registered first (top of the page on image above)
- Then customization part is registered by running Include: **includeCustomScheduleInit**. So, if there are any customizations added then they will be registered as well
- Then action: **actionStartScheduler** enables all the schedules



By clicking XEF helper (in the Modelyn top line on image above) you can open a list of XEF functions and filtering on “schedule” reflects a list of functions which can be used in relation to schedule customizations (see image above).

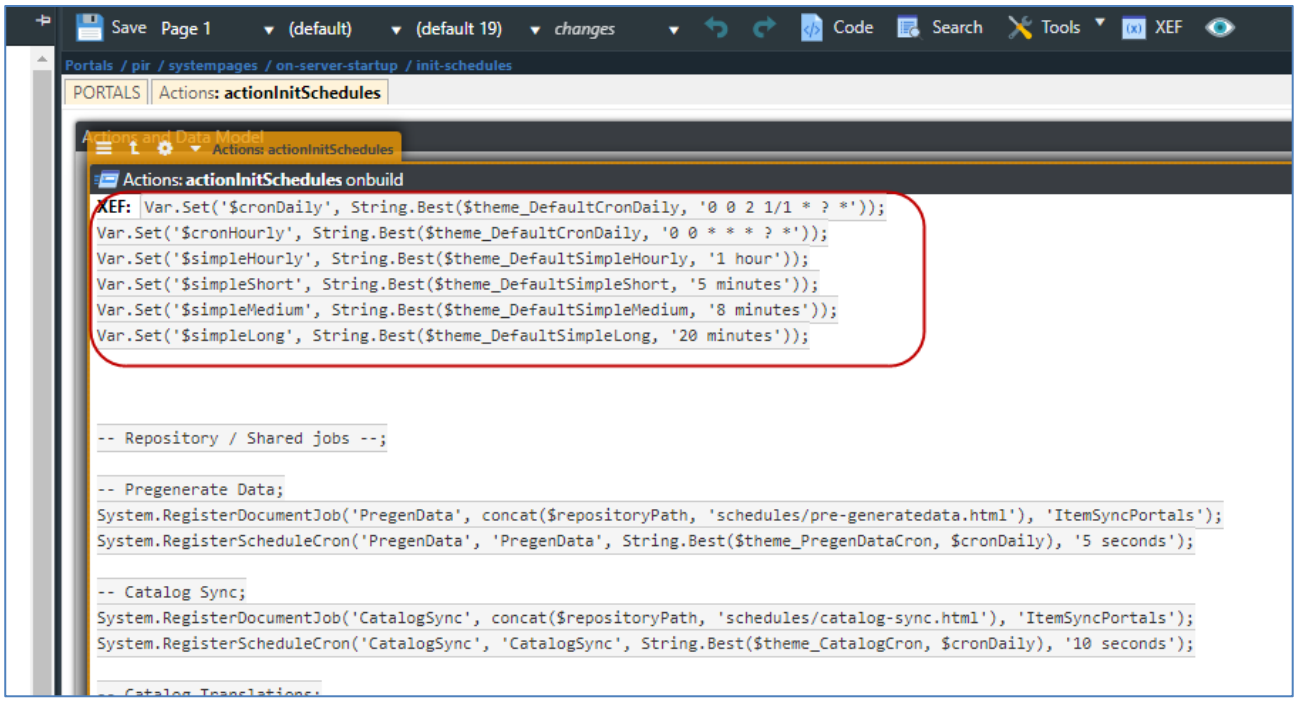
If schedules are registered correctly and website has been restarted, then a list of schedules should appear in ***System / Setup / Schedules***.



Note that if schedules are disabled in ***System / Setup / Schedules / Config*** and the website is rebooted, then schedules will be registered (as ***Content / Content Structure / pir / systempages / on-server-startup / init-schedules*** page runs every time website rebooted), but schedules will not be reflected in the list under ***System / Setup / Schedules.*** And schedules can only be run manually.

SCHEDULE REGISTRATION

The upper top section in action: actionInitSchedules declares the needed variables.



Then there comes a document, schedules cron and schedule simple registration part.

Document and schedule registration part

```
-- Repository / Shared jobs --;

-- Pregenerate Data;
System.RegisterDocumentJob('PregenData', concat($repositoryPath, 'schedules/pre-generatedata.html'), 'ItemSyncPortals');
System.RegisterScheduleCron('PregenData', 'PregenData', String.Best($theme_PregenDataCron, $cronDaily), '5 seconds');
```

DOCUMENT REGISTRATION:

```
System.RegisterDocumentJob('PregenData', concat($repositoryPath, 'schedules/pre-generatedata.html'), 'ItemSyncPortals');
```

Where:

`System.RegisterDocumentJob` - an XEF function

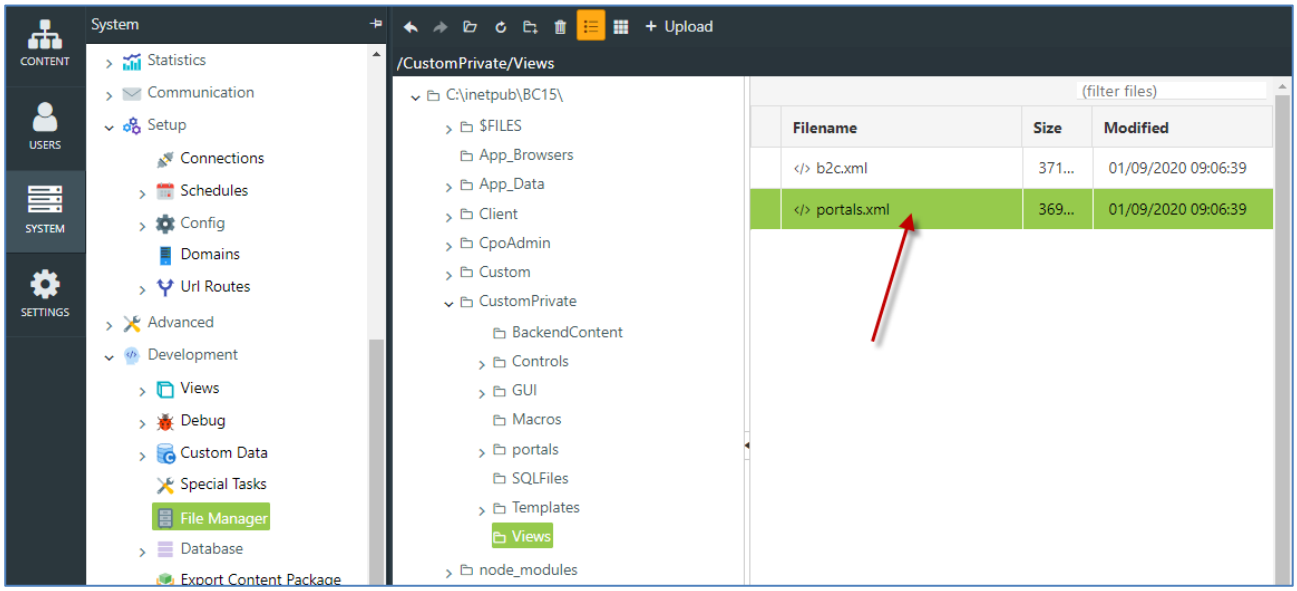
`PregenData` – job ID used for references between document and schedule. The job ID should reflect a purpose of the job

`concat($repositoryPath, 'schedules/pre-generatedata.html')` – document path. A path to a schedule document in a Modelyn content structure. "*concat*" function merges values given in

parentheses. `$repositoryPath` variable is used for B2B, SA and some common schedules. B2C schedule documents use variable `$b2cPath.`

```
59
60  -- Rebuild Indexes;
61  System.RegisterDocumentJob('RebuildIndexes', concat($repositoryPath, 'schedules/rebuild-indexes.html'), 'ItemSyr
62  System.RegisterScheduleCron('RebuildIndexes', 'RebuildIndexes', String.Best($theme_RebuildIndexesCron, $cronDail
63
64  -- NAV Keep Alive;
65  System.RegisterDocumentJob('NAVkeepAlive', concat($repositoryPath, 'schedules/nav-keep-alive.html'));
66  System.RegisterScheduleSimple('NAVkeepAlive', 'NAVkeepAlive', String.Best($theme_NavKeepAliveCron, $simpleMediun
67
68
69
70
71
72  -- B2C jobs --;
73
74  System.LogCustom('schedules', concat($b2cPath, 'systempages/schedules/callpregendata.html'));
75
76  -- Pregenerate Data;
77  System.RegisterDocumentJob('CallPregenData', concat($b2cPath, 'systempages/schedules/callpregendata.html'), 'Ite
78  -- Job registered so it can be called manually. But not scheduled here, as it is included in PopulateFilterData;
79
```

Both variables, **repositoryPath** and **b2cPath**, are declared and assigned in the view file **portals.xml** (under ***CustomPrivate/Views/portals.xml***) or can be found in Modelyn:



(double click on one of the .xml files to view the content)

The path is taken either from theme value **$theme_RepositoryPath** or from var **$repositoryPathDefault** (for B2C its theme value **$theme_B2CPath** or from var **$B2cPathDefault**), depending on what value is not empty, and the default value is declared one line above in the view file code and equals to "**/~/pir/**" in case with PIR (and "**/~/b2c/**" for B2C):

Each view file, ***b2c.xml*** and ***portals.xml***, has its own **RepositoryPath** theme value. The value in this theme value can be changed in ***Soft admin / Look book / Repository path***. It is by default set to "**/~/pir/**" and not displayed on front end.

```
342     <!-- Full cms path to promoted items templates -->
343     <zml:variable name="$promotedItemsTemplatesPath" select="concat($portalRoot, 'systempages/content/promoted-items-templates/')" />
344
345     <!-- Path to B2C (for media repository moving of custom content) -->
346     <!-- Note: we also now use $b2cPath in init-schedules page in PIR, where we init all schedules, b2c too -->
347     <zml:variable name="$b2cPathDefault" value="/~/b2c/" />
348     <!-- DGG: test for the schedule path!!!! -->
349     <!--<zml:variable name="$b2cSchedulesPath" select="String.Best($theme_B2cSchedulesPath, $b2cPathDefault)" />-->
350     <zml:variable name="$b2cPath" select="String.Best($theme_B2cPath, $b2cPathDefault)" />
351     <zml:variable name="$b2cVirtualShopPath" select="concat($b2cPath, 'shop')" />
352     <zml:variable name="$b2cProductsAbs" select="Url.MakeAbsolute($b2cPath, 'shop/products.html')" />
353     <zml:variable name="$b2cProductDetailsAbs" select="Url.MakeAbsolute($b2cPath, 'shop/details.html')" />
354
355
356     <!-- Path to B2B (for media repository link to product details page) -->
357     <zml:variable name="$b2bPathDefault" value="/~/b2b/" />
358     <zml:variable name="$b2bPath" select="String.Best($theme_B2bPath, $b2bPathDefault)" />
359
360     <!-- Path to SA (for media repository link to product details page) -->
361     <zml:variable name="$saPathDefault" value="/~/salesagent/" />
362     <zml:variable name="$saPath" select="String.Best($theme_SaPath, $saPathDefault)" />
363
364     <!-- Full cms path to repository -->
365     <zml:variable name="$repositoryPathDefault" value="/~/pir/" />
366     <zml:variable name="$repositoryPath" select="String.Best($theme_RepositoryPath, $repositoryPathDefault)" />
367
368     <!-- Full cms path to images folder (for additional product images) -->
369     <zml:variable name="$imagesPath" select="concat($repositoryPath, 'images/')" />
370
371     <!-- Full cms path to order-types folder (for order-type images) -->
372     <zml:variable name="$ordertypesPath" select="concat($repositoryPath, 'order-types/')" />
```

B2cPath and B2cPathDefault

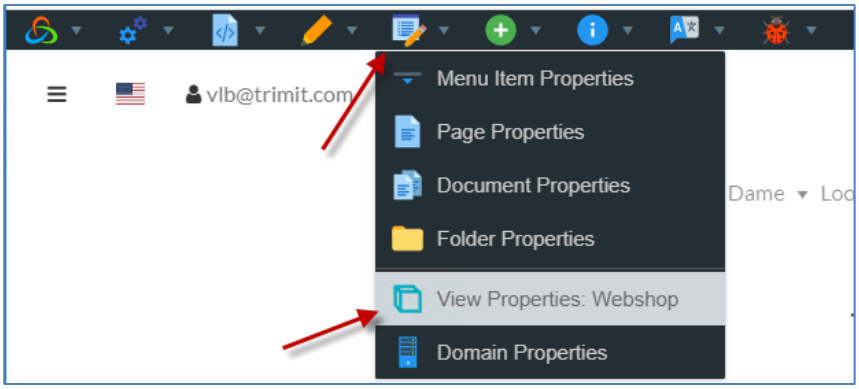
The path is taken either from theme value or from var repositoryPathDefault (or other portal) and the default is set one line above and equals "/~/pir/"

Theme values – **ReposigoryPath** theme value can be set both in soft admin and in theme value properties. Other theme values can be set only in theme value properties:

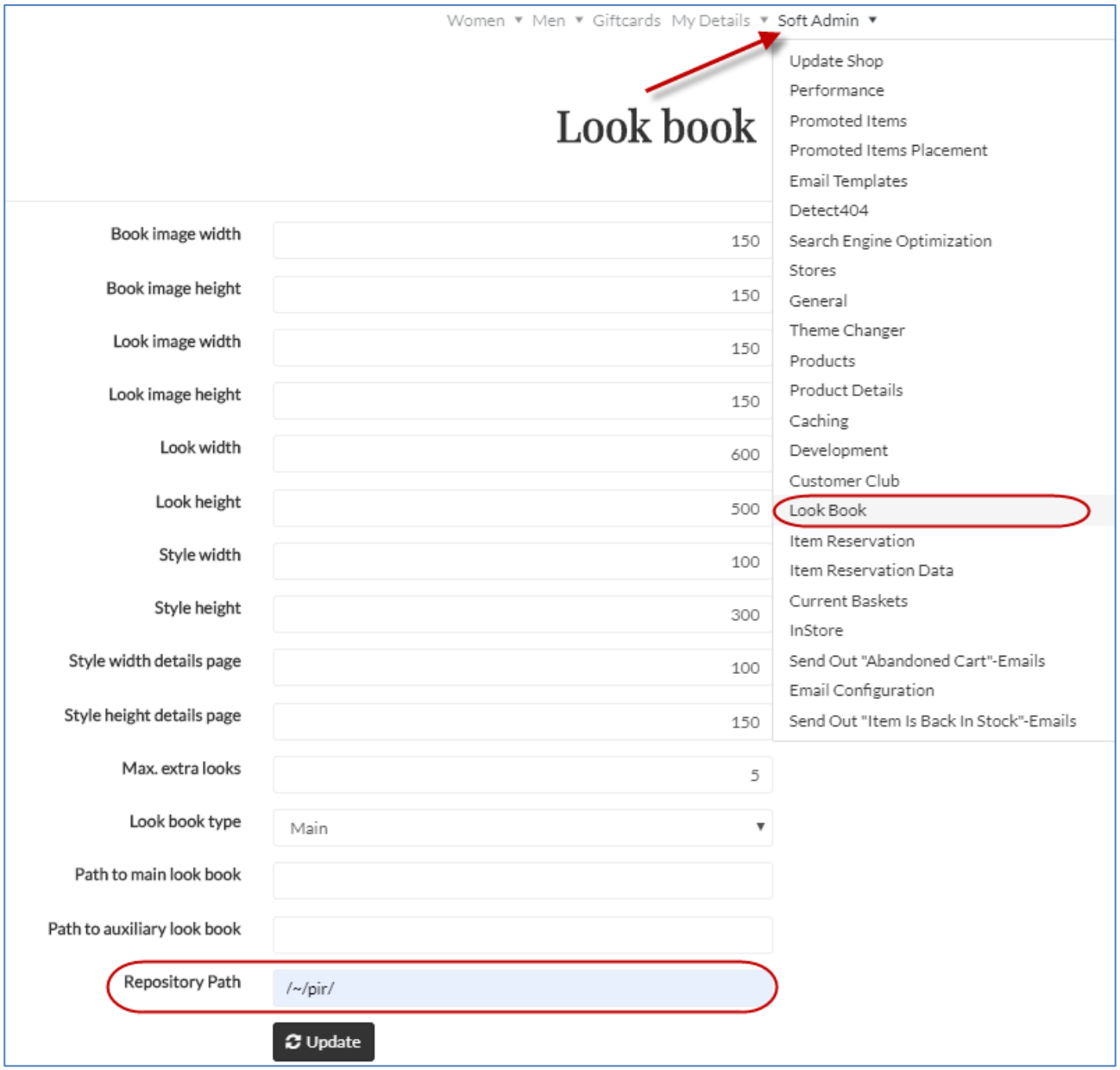
Save View Properties

```
32      <value id="AbandonedCartEmailContentBgColor">#ffffff</value>
33      <value id="AbandonedCartEmailFooterBgColor">#fafafa</value>
34      <value id="AbandonedCartEmailGoogleFontFamily">Montserrat</value>
35      <value id="AbandonedCartEmailButtonFontColor">#ffffff</value>
36      <value id="AbandonedCartEmailSiteRoot">http://b2c.bc13r2/</value>
37      <value id="AbandonedCartEmailB2cPath">https://b2c.bc13r2/b2c/</value>
38      <value id="ShowLookBookOverview">true</value>
39      <value id="RepositoryCrossDomainUrl">//pir.bc13r2/</value>
40      <value id="RepositoryPath">/~/pir/</value>
41      <value id="B2cPath">/~/b2c/</value>
42      <value id="B2bPath">/~/b2b/</value>
43      <value id="SaPath">/~/salesagent/</value>
44  </themevalues>
```

To open theme values – log in as admin and in Admin helper menu view properties:



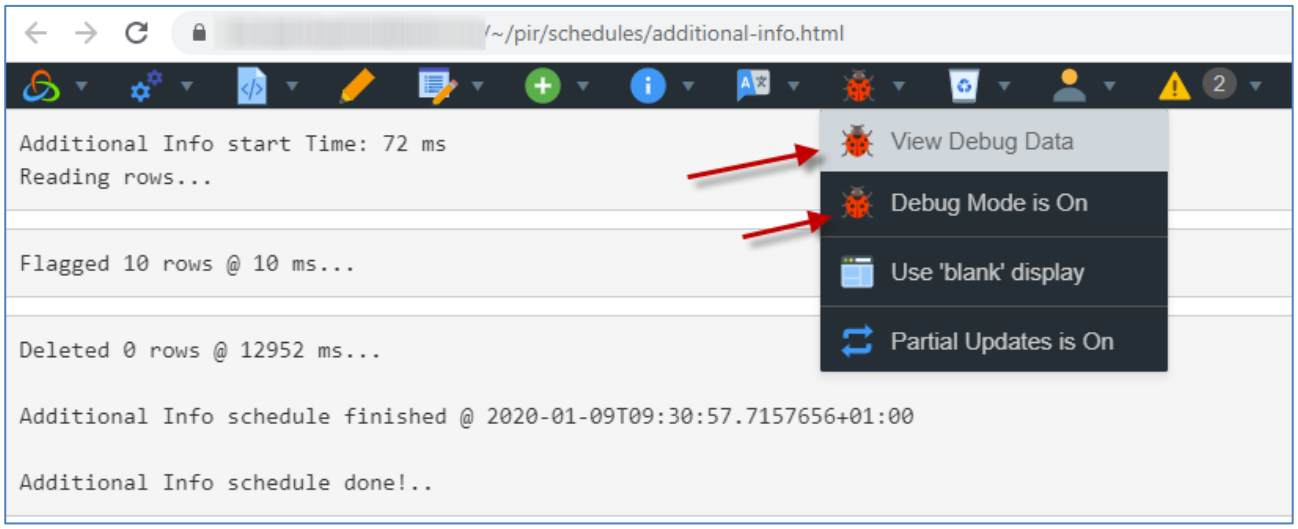
ReposigoryPath theme value can be set in soft admin:



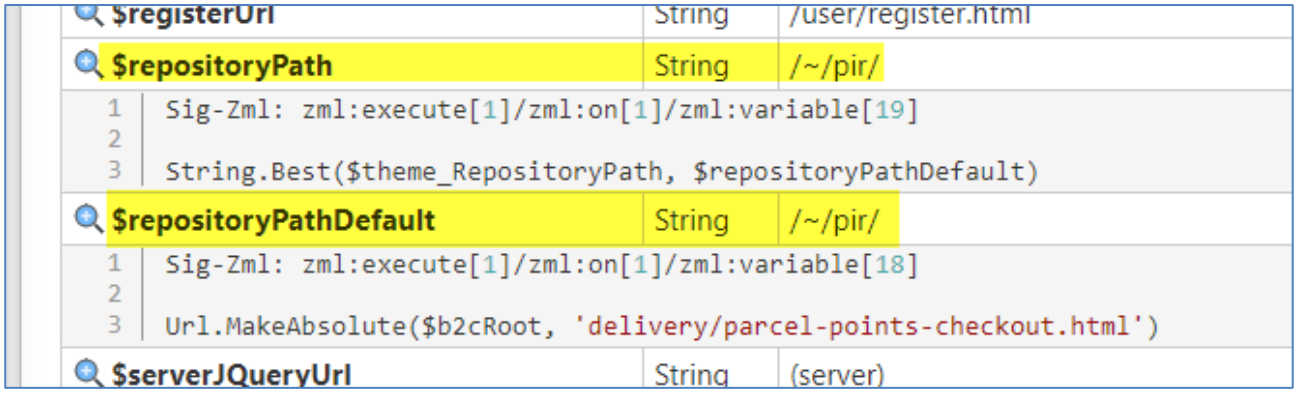
(the value of Repository Path is set in this example, but systems default value for Repository Path is also equal "/~/pir/", so on customer installations this field can be left blank)

Repository Path is also related to product image displaying on all portals, as images must be uploaded to PIR in case with B2C and can be uploaded to PIR in case with SA and B2B. Therefore, it is important to use a correct path here.

To check the values for `$repositoryPath or $repositoryPathDefault` variables, navigate to e.g. "additional info" schedule via this path in Modelyn ***Content / Content Structure / pir / schedules / additional info***. Right click on schedule and select View. New browser tab will open running schedule "additional-info" and check the debug data (make sure Debug Mode in ON):



Under Variables (scroll down), it is possible to see two variables: ***$repositoryPath*** and ***$repositoryPathDefault*** and their values.



Now back to schedule document registration parameters.

`ItemSyncPortals` - thread group, which allows running schedules in one group so that each next schedule will run only when the previous schedule is finished. Schedules with different thread group names will be executed disregarding the execution status of each other and in accordance with the time period specified. The thread group name value can be chosen freely (according to XML standards). For example: `threadgroup="single"`

Userid* and *password – file permission credentials if applicable.

SCHEDULE CRON REGISTRATION

```
System.RegisterScheduleCron('PregenData', 'PregenData', String.Best($theme_PregenDataCron, $cronDaily), '5 seconds');
```

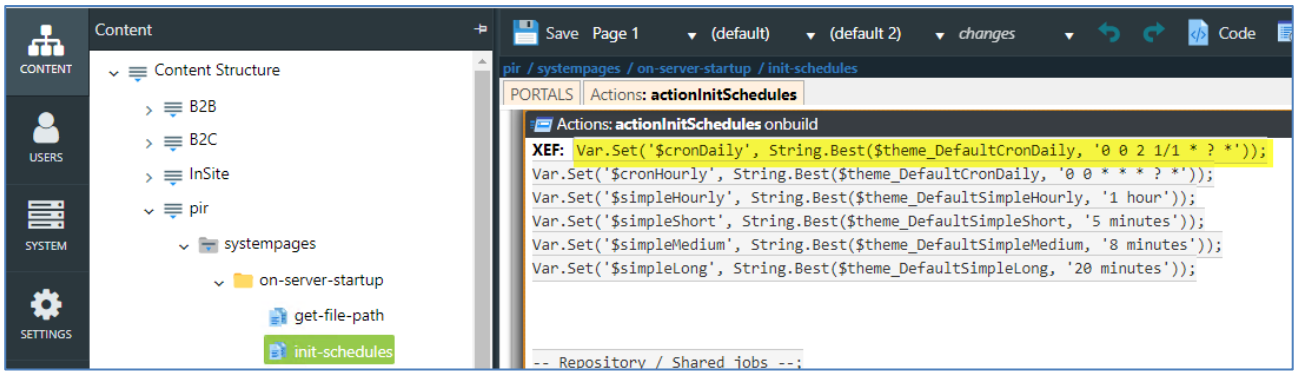
Where

`System.RegisterDocumentJob` – an XEF function

`PregenData` (1) – schedule ID

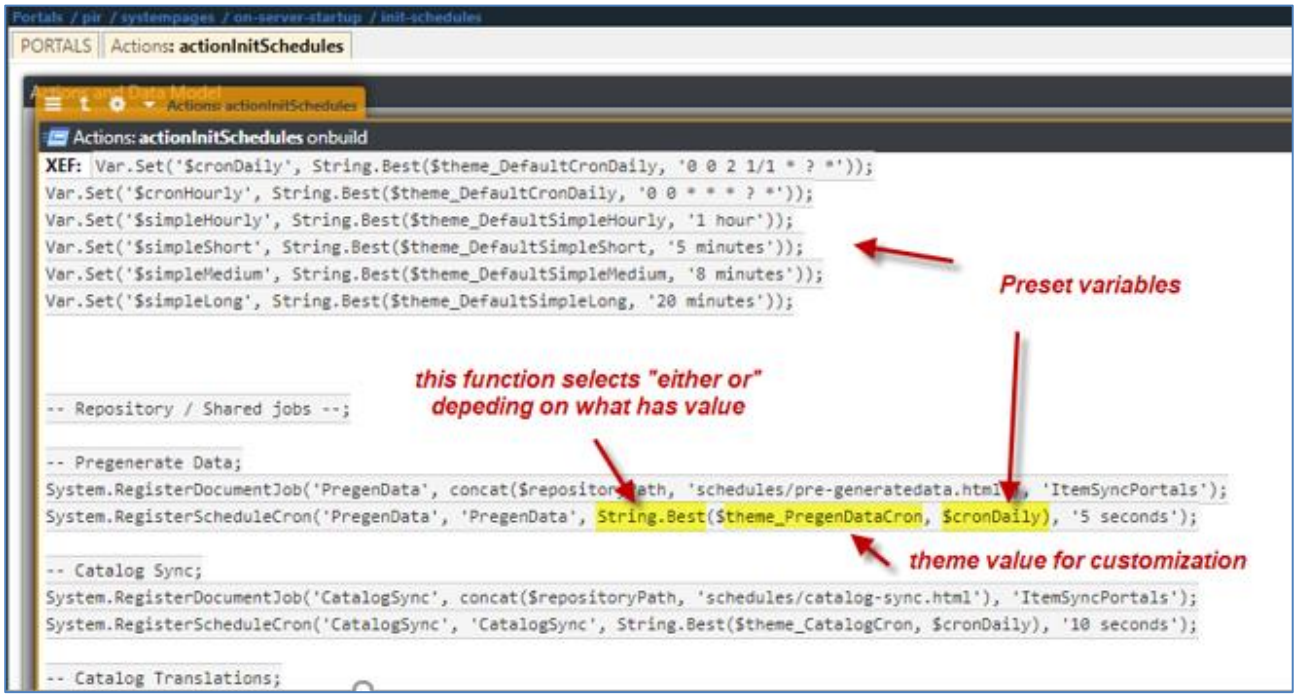
`PregenData` (2) – job ID

`String.Best($theme_PregenDataCron, $cronDaily)` – cron value taken either from **PregenDataCron** theme value or from **$cronDaily** variable, depending on what is not blank. **$cronDaily** variable is set by default to take value from theme value **DefaultCronValue** or, if theme value is not set, to be equal to '**0 0 2 1/1 * ? ***', which means 02.00 AM every day:



`'5 seconds'` – job execution delay before the first possible run.

Visual explanation:

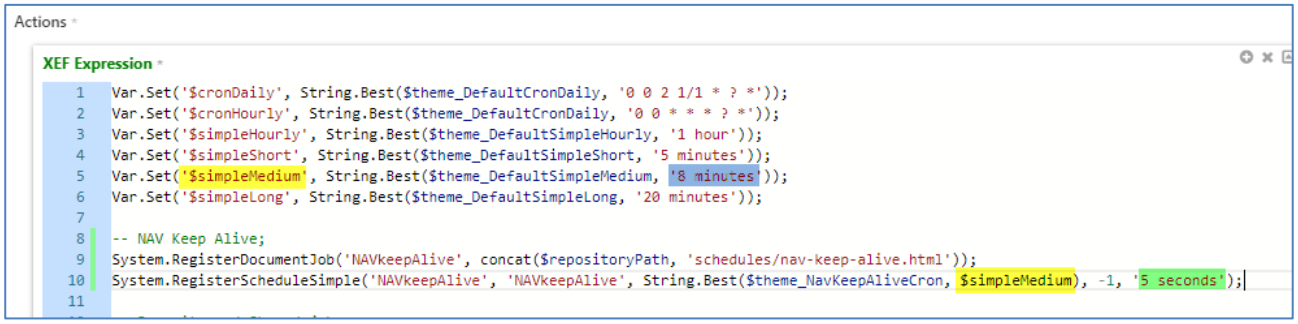


Schedule explanation

Schedule: specifies the job execution frequency and delay before the first possible run.

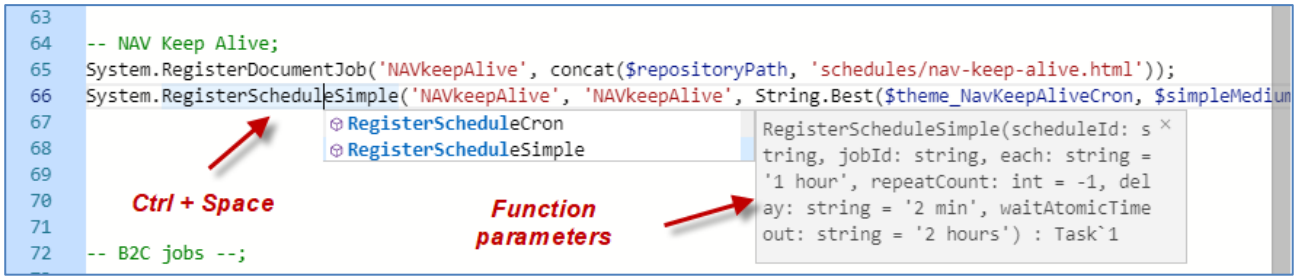
Schedules can be executed in two different ways: **simple** and **cron**

Simple: A simple trigger will execute by period of *10 minutes*, *1 hour*, *10 seconds,* etc.
In this example schedule "NAVKeepAlive" is set to be executed each 8 minutes (if theme value "DefaultSimpleMedium" is blank) with 5 seconds delay:



"-1" in simple schedule registration line (last line in in picture above) means a repeat count, i.e. how many times to repeat, it this case - do not repeat.

In order to find out about XEF function parameters, open Modelyn and navigate to ***Content / Content structure/ pir/ systempages / on-server-startup /*** and open code for ***init-schedules***. Double click on actionInitSchedules area. In Actions under XEF Expressions, put a cursor on the function and click *Ctrl + Space* (eventually click twice), then use arrow Up or Down to select the needed function – the side info box will reflect a parameter description:



Cron: Cron schedule (http://en.wikipedia.org/wiki/Cron) is more advanced, but allows e.g. a trigger that fires at 10:30, 11:30, 12:30, and 13:30, on every Wednesday and Friday by entering the following expression: "0 30 10-13 ? * WED,FRI".

In the following example schedule is set to be executed every day at 02.00 AM with 5 seconds delay:

Actions

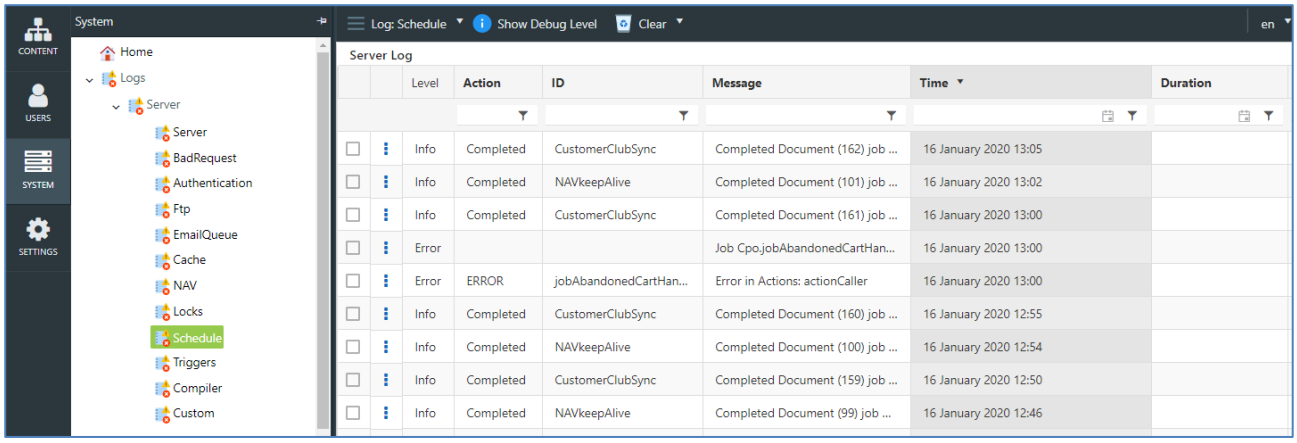
XEF Expression

```
Var.Set('$cronDaily', String.Best($theme_DefaultCronDaily, '0 0 2 1/1 * ? *'));
Var.Set('$cronHourly', String.Best($theme_DefaultCronDaily, '0 0 * * * ? *'));
Var.Set('$simpleHourly', String.Best($theme_DefaultSimpleHourly, '1 hour'));
Var.Set('$simpleShort', String.Best($theme_DefaultSimpleShort, '5 minutes'));
Var.Set('$simpleMedium', String.Best($theme_DefaultSimpleMedium, '8 minutes'));
Var.Set('$simpleLong', String.Best($theme_DefaultSimpleLong, '20 minutes'));



-- Repository / Shared jobs --;

-- Pregenerate Data;
System.RegisterDocumentJob('PregenData', concat($repositoryPath, 'schedules/pre-generatedata.html'), 'ItemSyncPortals');
System.RegisterScheduleCron('PregenData', 'PregenData', String.Best($theme_PregenDataCron, $cronDaily), '5 seconds');

```

For more **Cron** expressions visit the following link:
http://www.cronmaker.com/

The placement of the **Cron** schedule elements is important – they have to be placed after documents (if they are activated).

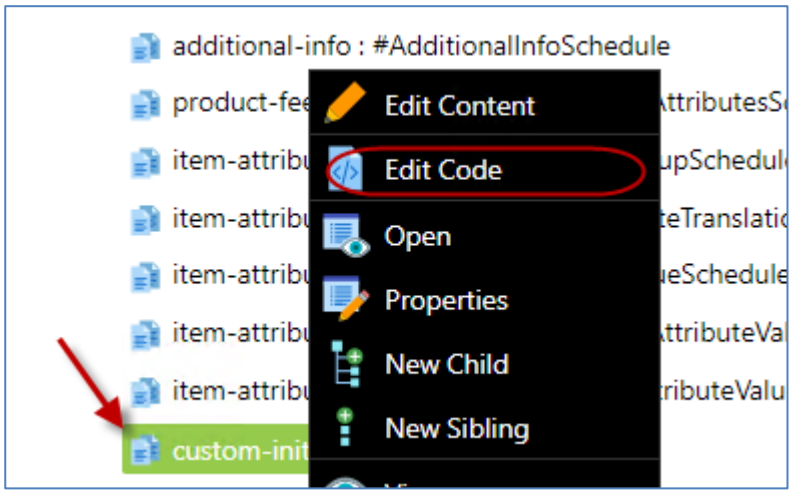
All the cron initiated schedule runs are logged and can be found under
Modelyn/System/Logs/Server/Schedule



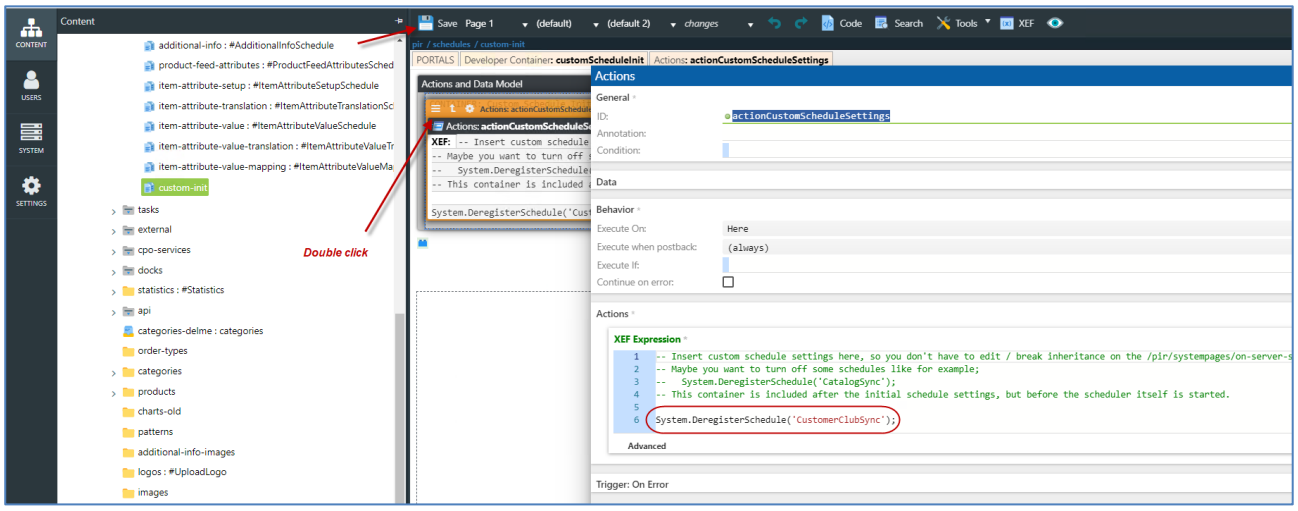
NOTE:
If a Cron expression is set to run a job at a specific hour – be aware that this is a UTC time, so it can be different from the actual time and conversion would be needed (for example, 12.00 o'clock in UTC (Cron: "0 0 12 1/1 * ? *") - in United Kingdom equals to 13.00 in Western Europe).
For UTC time conversion visit: http://www.worldtimeserver.com/convert_time_in_UTC.aspx

SCHEDULE CUSTOMIZATIONS

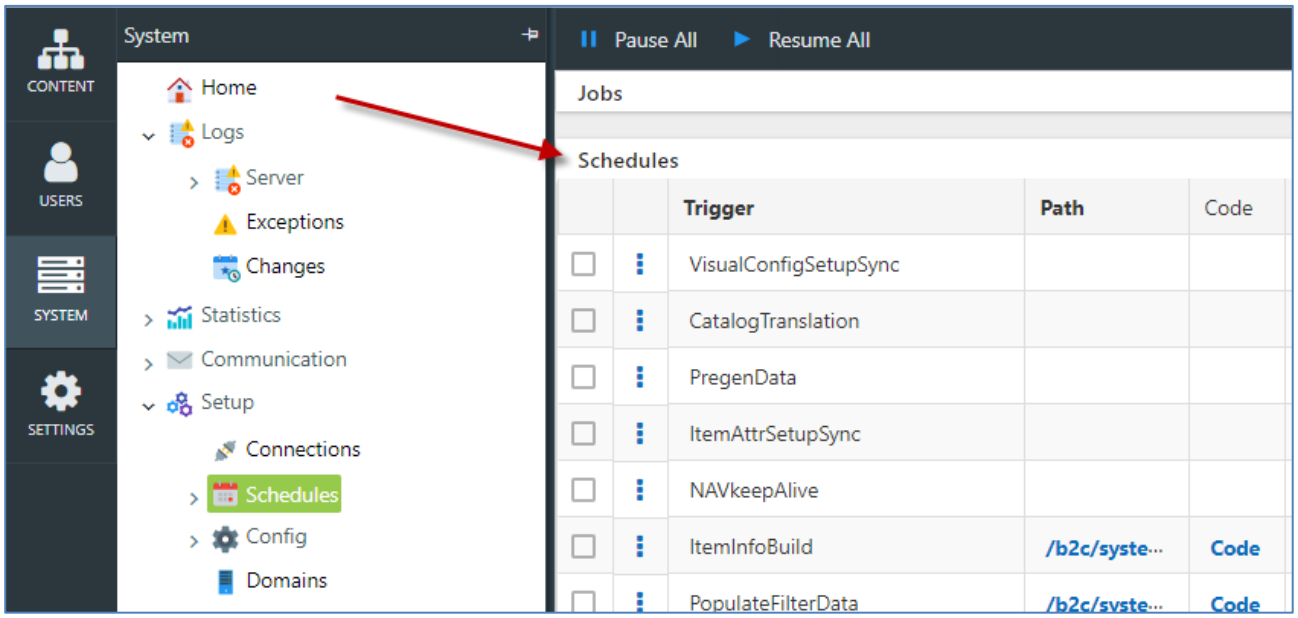
If schedule has to be customized, it has to be done in Modelyn under ***Content / Content Structure / pir / schedules / custom-init***. Right click and select Edit Code.



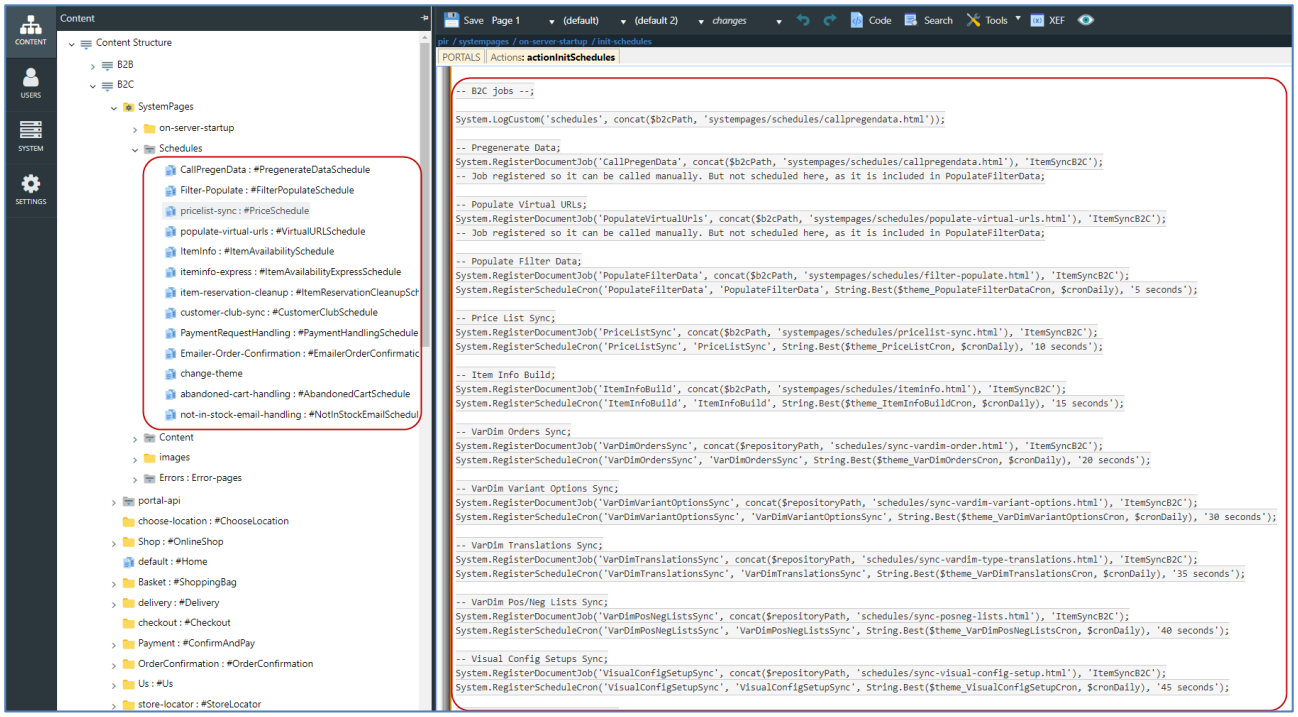
Here double click on action actionCustomScheduleSetting. In XEF expression section add needed customization:



In the XEF function use schedule ID in parenthesis. When finished, click **Save**. After each customization it is required to reboot the application pool in order for changes to take effect. If schedule was deregistered, then after application pool restart, the schedule should be gone from a list of registered schedules under ***System / Setup / Schedules***:


CONTENT
USERS
SYSTEM
SETTINGS
System
Home
Logs
Server
Exceptions
Changes
Statistics
Communication
Setup
Connections
Schedules
Config
Domains
Pause All
Resume All
Jobs
Schedules
Trigger
Path
Code
VisualConfigSetupSync
CatalogTranslation
PregenData
ItemAttrSetupSync
NAVkeepAlive
ItemInfoBuild
/b2c/syste···
Code
PopulateFilterData
/b2c/syste···
Code

DESCRIPTION OF SCHEDULES (B2C WEBSHOP)



PREGENERATE DATA SCHEDULE

*(**Default view**: CallPregenData.html; **'Default:Title' view:** CallPregenData : #PregenerateDataSchedule)*
Runs a pre-data generation of the categories with all Items within and saving it to buffer table in TRIMIT. So, no data is sent to Modelyn. The reason is to speed up any communication with the category structure. The schedule frame is equivalent to the changes done in TRIMIT. So, if changes in TRIMIT are made often then CallPregenData should also be run often. If changes in TRIMIT are made rarely, then there is no need to run this Schedule often as well.

ITEM SCHEDULE

*(**Default view**: FilterPopulate.html; **'Default:Title' view**: FilterPopulate : #FilterPopulateSchedule)*
This schedule uses the pre-generated category structure to find all Items, Colors, Sizes and Language Texts needed for the Webshop/Portal. Run this under consideration to how much the Items change in the Category, Colors disabled, etc.
This schedule includes **CallPregenData** Schedule, this means that when **Filter-Populate** Schedule is running it also initiates **CallPregenData** Schedule.
Running this schedule right after the B2C Webshop installation also creates master folders in Modelyn under ***Content/Content Structure/pir/products***.

PRICE SCHEDULE

*(**Default view**: pricelist-sync.html; **'Default:Title' view**: pricelist-sync : #PriceSchedule)*
This schedule updates all Item prices for all currencies activated in the back end (TRIMIT) including normal and campaign prices together with variant specific prices. It is possible to run this schedule manually from the B2C Webshop soft admin.

VIRTUAL URL SCHEDULE

*(**Default view**: populate-virtual-urls.html; **'Default:Title' view**: populate-virtual-urls : #VirtualURLSchedule)*

This schedule searches the filter data and maps the URLs enabling the system to the nice URLs. **PopulateVirtualUrls** needs to be run on server start so that there are no 404s, but it is also run as part of **Filter-Populate** (called from within the schedule)

ITEMAVAILABILITY SCHEDULE

*(**Default view**: ItemInfo.html; **'Default:Title' view**: ItemInfo : #ItemAvailabilitySchedule)*
This is a reduced run under the same conditions as the **Item Schedule** (FilterPopulate) – but is there to maintain the item availability – this should run frequently to update the Item availability on the portal. Remember that the availability check is real time so out of synch, you cannot sell Item non available anyway – but the info on the Item can be misleading in periods of time.
In connection to Item Reservation Functionality (in B2C Webshop) – this Schedule, when running, cleans up all the purchased and outdated Item Reservations.

ITEM AVAILABILITY EXPRESS SCHEDULE

*(**Default view**: iteminfo-express.html; **'Default:Title' view**: iteminfo-express : #ItemAvailabilityExpressSchedule)*
This schedule is a small edition of **Item Availability Schedule** (ItemInfo) with a difference that this Schedule runs more frequently and aimed for already purchased Items specifically.
Here is how it works (B2C Webshop with item reservation):
After an Item has been purchased then the Item Reservation record with this Item gets a checkmark in field **Purchased** and the record time stamp is updated, all other checkmarks are cleared out.
iteminfo-express schedule runs frequently, i.e. every 5 minutes, and scans all Item Reservation records to find the purchased ones. As soon as the time stamp of the purchased record becomes older than 15 minutes (this can be adjusted in soft admin) – ***iteminfo-express*** schedule synchronizes the availability of the specific Item with TRIMIT and deletes the purchased record from the Item Reservation.

NOTE:
Be aware that this schedule, *ItemInfo-Express*, should only be enabled when Item Reservation is enabled on the B2C Webshop.

ITEM RESERVATION CLEANUP SCHEDULE

*(**Default view**: item-reservation-cleanup.html; **'Default:Title' view**: item-reservation-cleanup : #ItemReservationCleanupSchedule)*
This Schedule cleans up expired reservations (in **CustomData** table, group "TRM-RESV") when "***Max. reservation time***" is reached (this value is adjustable in soft admin) and also deletes Item Reservations for no longer online user sessions. The frequency of running this Schedule has to be high, i.e. every 5 minutes.

NOTE:
Item Reservation Cleanup schedule should always be enabled even though item reservation is not used on customers B2C webshop. It should run e.g. once in 24 hours. Item reservation data in Modelyn database is used to calculate availability for the current user session and to prevent oversell. This approach is used to avoid a round-trip to TRIMIT.

CUSTOMER CLUB SCHEDULE

*(**Default view**: customer-club-sync.html; **'Default:Title' view**: customer-club-sync : #CustomerClubSchedule)*
This schedule synchronizes the Customer Club Levels between TRIMIT and Modelyn. After run of this schedule Customer Club Levels of the Modelyn users will be updated.

PAYMENT HANDLING SCHEDULE

*(**Default view**: PaymentRequestHandling.html; **'Default:Title' view**: PaymentRequestHandling: #PaymentHandlingSchedule)*
This schedule handles capture/refunds/cancellation of payments. It should run by the minute. It is a lightweight schedule for exchanging data from TRIMIT and has a little performance cost. If auto capture etc. is not in the customer setup then this schedule can be ignored.

EMAILER ORDER CONFIRMATION SCHEDULE

*(**Default view**: Emailer-Order-Confirmation.html; **'Default:Title' view**: Emailer-Order-Confirmation: #EmailerOrderConfirmationSchedule)*
Calls TRIMIT for E-Mails – it should run by the minute serving customers with E-Mail Order Confirmation and what else is passed in the queue.
This is a legacy schedule and no longer active in the Modelyn.

CHANGE THEME SCHEDULE

*(**Default view**: change-theme.html)*
This schedule helps automatically enabling theme changes made by admin.
Not implemented in this version.

ABANDONED CART HANDLING SCHEDULE

*(**Default view**: abandoned-cart-handling.html; **'Default:Title' view**: abandoned-cart-handling: #AbandonedCartSchedule)*
When enabled from B2C soft admin, this schedule sends an email with a reminder to those registered users who have added an item to basket in their previous user sessions. At the same time older baskets can be set to deletion by this schedule.

NOT IN STOCK EMAIL HANDLING SCHEDULE

*(**Default view**: not-in-stock-email-handling.html; **'Default:Title' view**: not-in-stock-email-handling: #NotInStockEmailSchedule)*
This schedule sends an email to a B2C webshop user in case of an item this user was previously interested in (by signing up for email notification) when product is back in stock. The schedule also deletes a "user request" record from the Modelyn table, that is why the user notification email is sent only once.

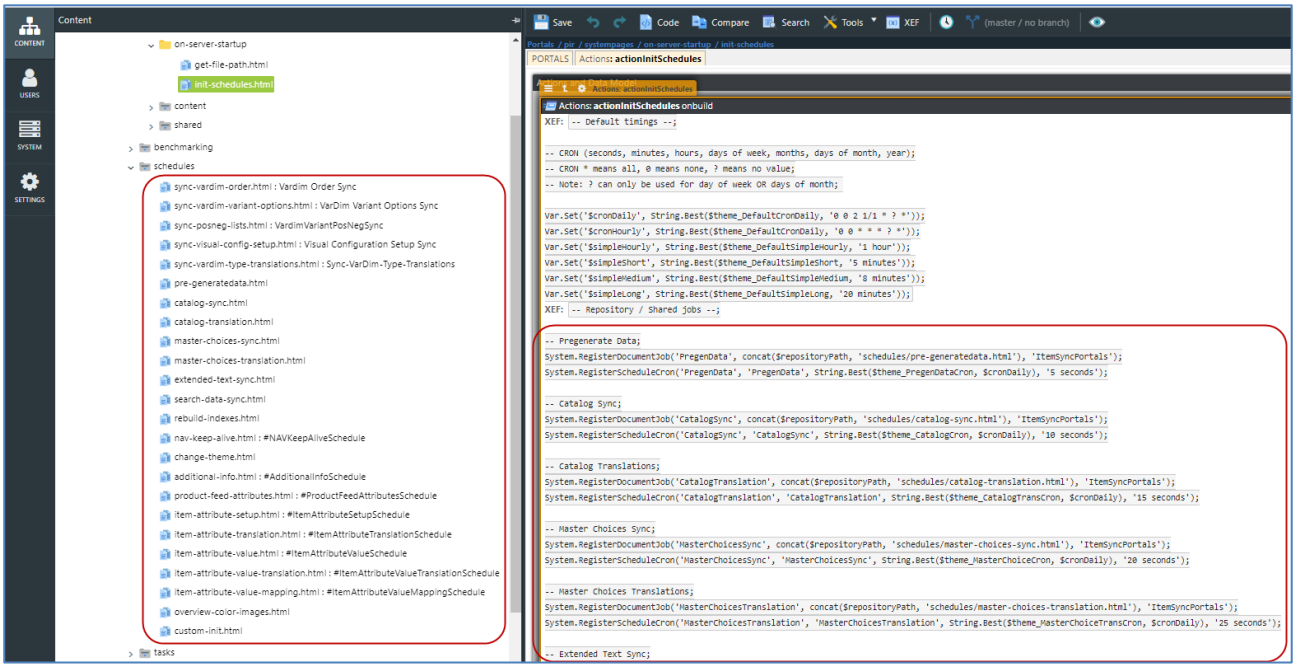
SEARCH DATA SYNC

Related to visual drop-down search on SA portal and B2B and B2C webshops. Data is synched to Custom data table in Modelyn, TRM-SEARCH-DATA group.

CATALOG-SYNC

Running this schedule in B2C webshop is needed to fetch data from TRIMIT and use it for configurable products.

DESCRIPTION OF SCHEDULES (SA, B2B AND GENERAL)



SYNC-VARDIM-ORDER

Synchronizes visual configuration master data for B2C webshop

SYNC-VARDIM-VARIANT-OPTIONS

Synchronizes visual configuration master data for B2C webshop

SYNC-POSNEG-LIST

Synchronizes visual configuration master data for B2C webshop

SYNC-VISUAL-CONFIG-SETUP

Synchronizes visual configuration master data for B2C webshop

SYNC-VARDIM-TYPE-TRANSLATIONS

Synchronizes visual configuration master data for B2C webshop

PRE-GENERATEDATA

Runs generation of the Category Structures, it forces activation of Category buffer in TRIMIT

CATALOG-SYNC

Synchronizes what we see inside the Categories (Custom Data->TRM_CATALOG), all the base data like masters and flat Items. Running this schedule right after the SA portals or B2B Webshop installation also creates master folders in Modelyn under ***Content/Content Structure/pir/products***.

NOTE:
Prices are not synchronized on SA portal and B2B Webshop but taken directly from TRIMIT and cached.

CATALOG-TRANSLATION-SYNC

Everything before it is synchronized in base (neutral) language from TRIMIT.

Identifies colors and item translations

MASTER-CHOICES-SYNC

Builds full variant list for each Item, description of the Item, colors and sizes

MASTER-CHOICES-TRANSLATION

Translates all the variants for each Item (it is not possible to search for color and variant until you ran the Translation synchronization)

EXTENDED-TEXT-SYNC

This schedule synchronizes all extended texts for the items in the catalog. Can be called with *?queryitem=<product number>* to only sync that master.

ITEM ATTRIBUTE SETUP

*(**Default view**: item-attribute-setup.html; **'Default:Title' view**: item-attribute-setup: #ItemAttributeSetupSchedule)*

Synchronizes records from ***Portal Item Attribute Management*** table in BC (attribute names). Data is synched to Custom data, TRM-ITEM-ATTRIBUTE-SETUP group.

ITEM ATTRIBUTE TRANSLATION

*(**Default view**: item-attribute-translation.html; **'Default:Title' view**: item-attribute-translation: #ItemAttributeTranslationSchedule)*

Synchronizes attribute name transaltions. Data is synched to Custom data, TRM-ITEM-ATTRIBUTE-TRANSLATION group.

ITEM ATTRIBUTE VALUE

*(**Default view**: item-attribute-value.html; **'Default:Title' view**: item-attribute-value: #ItemAttributeValueSchedule)*

Synchronizes item attributes values from table ***Item Attribute Values*** in BC. For example, if item attribute name is Age Group then attribute values could be: *newborn*, *kids*, *adult* etc. Data is synched to Custom data, TRM-ITEM-ATTRIBUTE-VALUE group.

ITEM ATTRIBUTE VALUE MAPPING

*(**Default view**: item-attribute-value-mapping.html; **'Default:Title' view**: item-attribute-value-mapping: #ItemAttributeValueMappingSchedule)*

Synchronizes attribute name, value and related item number. Data is synched to Custom data, TRM-ITEM-ATTRIBUTE-VALUE-MAP group.

ITEM ATTRIBUTE VALUE TRANSLATION

*(**Default view**: item-attribute-value-translation.html; **'Default:Title' view**: item-attribute-value-translation: #ItemAttributeValueTranslationSchedule)*

Synchronizes attribute value translations. Data is synched to Custom data, TRM-ITEM-ATTRIBUTE-VALUE-TRANS group.

PRODUCT FEED ATTRIBUTES

*(**Default view**: product-feed-attributes.html; **'Default:Title' view**: product-feed-attributes: #ProductFeedAttributesSchedule)*

Related to Google Shopping and Facebook Pixel. Synchronized data is based on setup in ***Portal Setup*** / ***Data Feed Setup*** fast tab in BC. Data is synched to Custom data, TRM-FEED-ATTRIBUTES group.

SEARCH DATA SYNC

Related to visual drop-down search on SA portal and B2B and B2C webshops. Data is synched to Custom data, TRM-SEARCH-DATA group.

REBUILD-INDEXES

Rebuilds table indexes in Modelyn database, which helps on performance. More information on this Schedule read further in this document.

NAV-KEEP-ALIVE SCHEDULE

*(**Default view**: nav-keep-alive.html; **'Default:Title' view**: nav-keep-alive: #NAVKeepAliveSchedule)*
This is a simple keep alive schedule, ensures that the Business Central Web Service does not enter a sleep or connection/user license dispose.

ADDITIONAL INFO SYNC

Synchronizes care label and composition product information on SA, B2B and B2C. Data is synched to Custom data, TRM-ADD-INFO group.

OVERVIEW COLOR IMAGES

Runs through TRM-REP-MASTERS to detect and save changes in regards to product overview color images on B2C.

CUSTOM INIT

This schedule document is reserved for any customizations like re register/ register, changing schedule running frequency etc. This schedule document is executed on website restart (or server restart or application pool restart).

INDEX REBUILD SCHEDULE

The main purpose of this Schedule is to increase the performance of fetching data within the Modelyn database.
This Schedule initiates rebuilding indexes in the following Modelyn tables:

- ***CustomData***
- ***Domains***
- ***Documents***
- ***UserGroups***
- ***UserGroupMembers***
- ***Menus***
- ***Stats***
- ***StatLog***
- ***Files***

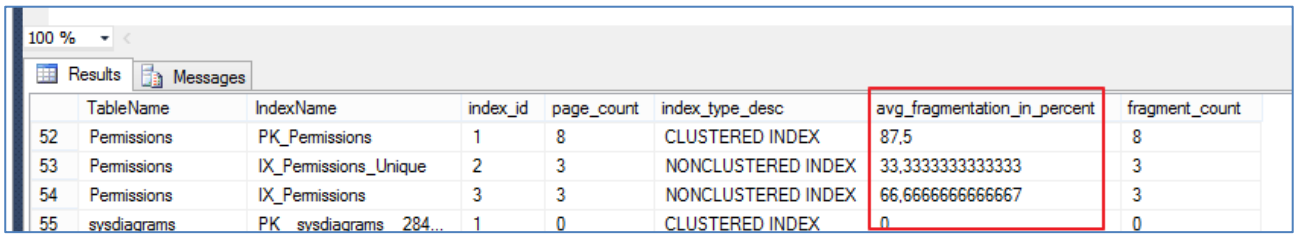
- ***Permissions***
- ***PermissionLinks***
- ***Pages***
- ***Users***
- ***Files***.

This schedule resides under ***Pir/schedules*** in Modelyn content structure. It should not be running very often, for example once a month or so. Possible cron realization is as follows: "*0 0 2 ? 1/1 SUN#1 **".
It does not take too many server- or Modelyn resources to run this Schedule.

It is also possible to see an average fragmentation in percent running the following script (for advanced users):

```
SELECT
    OBJECT_NAME(A.[object_id]) as 'TableName',
    B.[name] as 'IndexName',
    A.[index_id],
    A.[page_count],
    A.[index_type_desc],
    A.[avg_fragmentation_in_percent],
    A.[fragment_count]
FROM
    sys.dm_db_index_physical_stats(db_id(),NULL,NULL,NULL,'LIMITED') A INNER JOIN
    sys.indexes B ON A.[object_id] = B.[object_id] and A.index_id = B.index_id
```

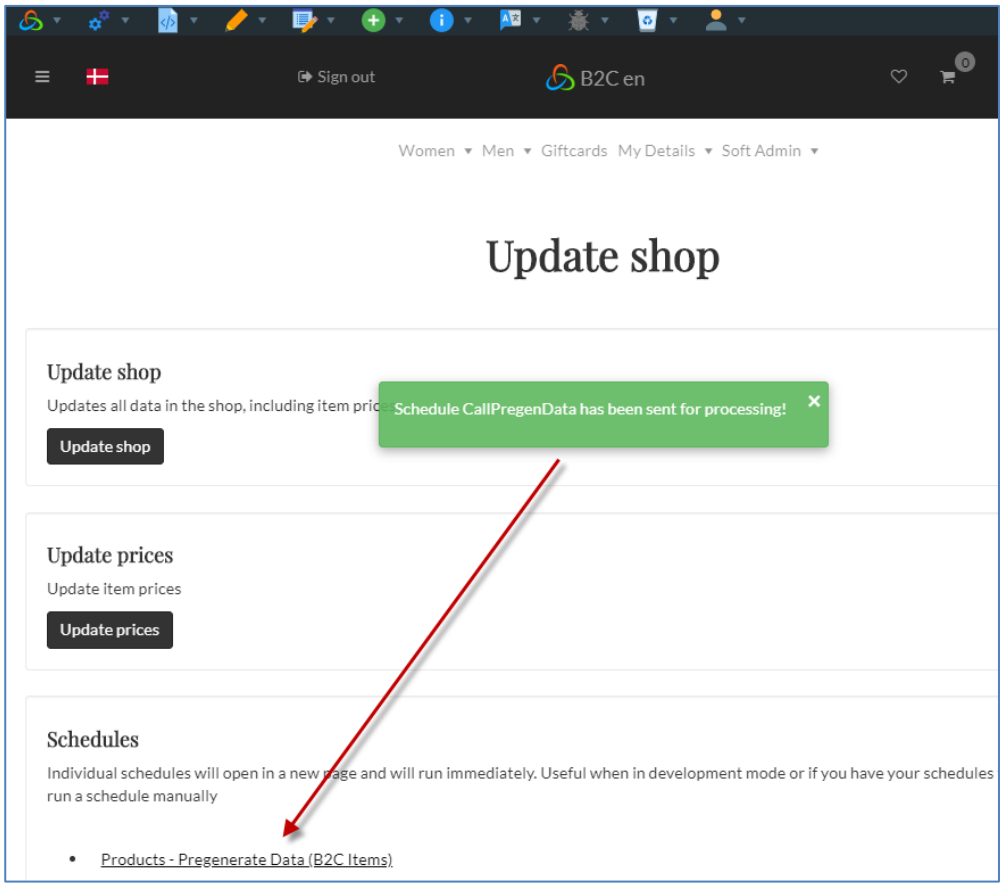
The result:



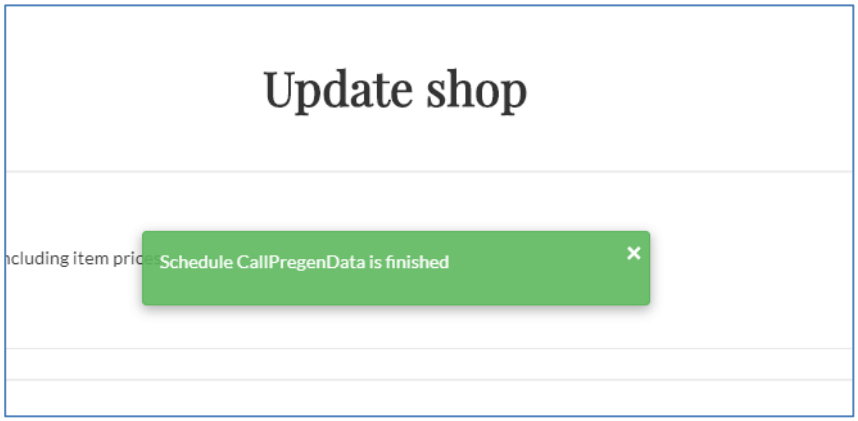
	TableName	IndexName	index_id	page_count	index_type_desc	avg_fragmentation_in_percent	fragment_count
52	Permissions	PK_Permissions	1	8	CLUSTERED INDEX	87,5	8
53	Permissions	IX_Permissions_Unique	2	3	NONCLUSTERED INDEX	33,3333333333333	3
54	Permissions	IX_Permissions	3	3	NONCLUSTERED INDEX	66,6666666666667	3
55	sysdiagrams	PK_sysdiagrams_284...	1	0	CLUSTERED INDEX	0	0

SCHEDULE RUN STATUS

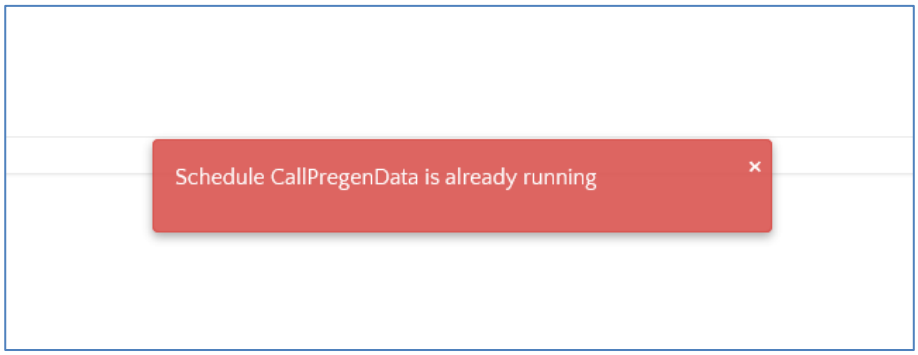
When running schedules manually it may be difficult to see if the schedule is still running and if the schedule has finished the synchronization. The front end can notify user about schedule start and schedule end when user navigates or refreshes portals pages.
Schedule initiated client notification on the front end:



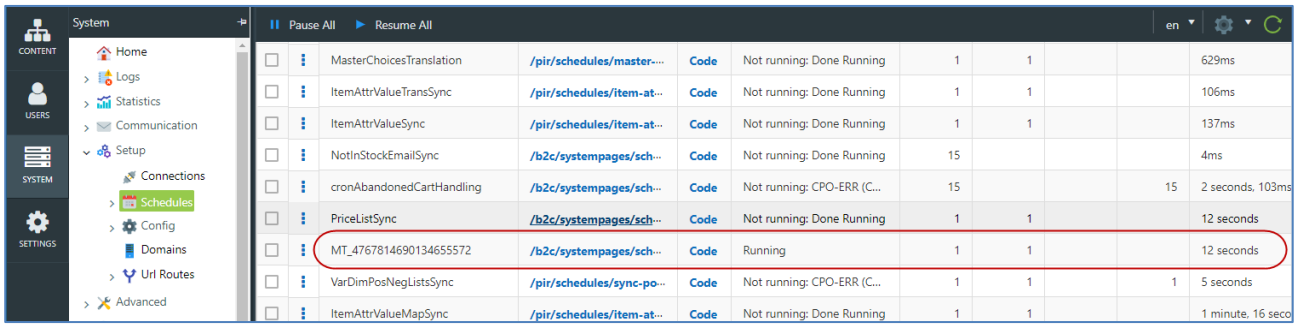
Schedule completed client notification (appears when schedule run is finished and while user navigates the portal or refreshes the page (F5)):



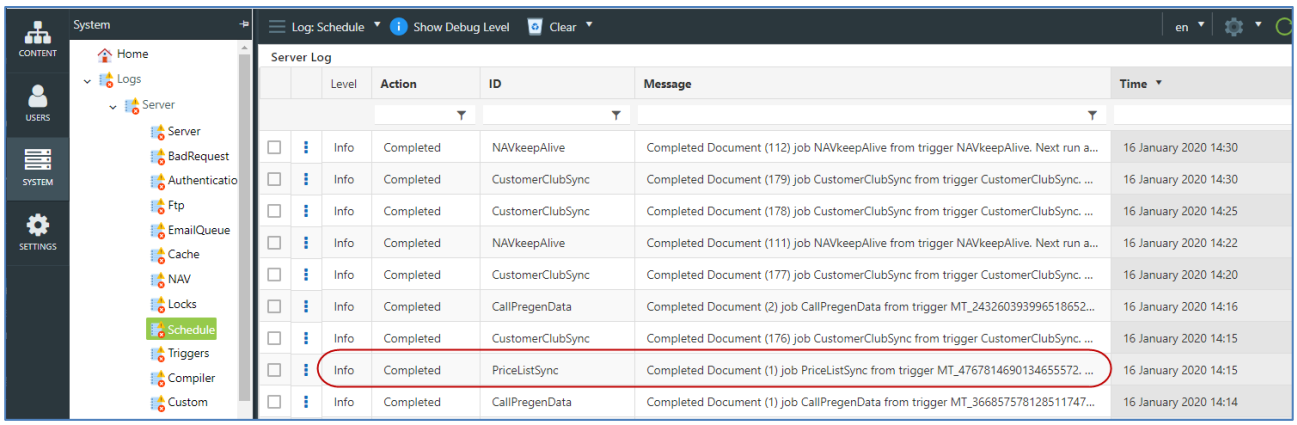
Client notification about schedule has already been initiated (initiating the same schedule twice):



Schedule run can take a long time, because of a long category structure or other reasons. Therefore, in order to see the schedules run status it is needed to go to Modelyn admin and navigate to ***System / Setup / Schedules***. Normally, this page contains a list of registered schedules set to run automatically, with help of cron. The schedules which are currently running are also displayed here. The schedules which are initiated manually are marked with "MT_*<random long number>*", where "MT" stands for manual trigger.
Running schedule initiated manually:



When the schedule run is completed then the status is displayed under ***System / Logs / Server / Schedule***. The schedule initiated manually will be marked in the same manner, i.e. "MT_*<random long number>*". If the schedule synchronization was completed successfully then the value in column ***Action*** will be equal to *Completed*. If the schedule failed to run, then value in ***Action*** column will be set to *Error* and column ***Message*** will have an error message.
Finished schedule initiated manually:



NOTE:
If schedules were disabled by setting <triggers disabled="true"/> in schedules config (can be found in ***Modelyn / System / Setup / Schedules / Config***) – it is required to reboot the website.
Then schedules will not be running automaticity according to simple or cron expressions. It will only be possible to run schedules manually either by activating **Update Shop** or running schedules individually.