SEO: Page Meta Tag Rules FAQ

The purpose of this document is to enable the internal stakeholders with details related to Page Meta Tag Rules:

Page Meta Tag Rule Overview

What are Page Meta Tag Rules?

Page meta tags are used in HTML markup to provide structured data about a web page to Search Engine and Social media bots. Common tags are title, robots, description, and social tags such as open graph tags. Depending on the number of localized pages and used meta tags, SEO teams might need to manage a lot of content to provide meaningful data for those tags. The Page Meta Tag Rules capability allows the generation and management of corresponding meta tag content on a large scale. Therefore, the Business Manager now includes a rule-based capability to create meta data for home, product (detail and listing) and content (detail and listing) pages.

What does Page Meta Tag Rules Offer to the Commerce Cloud merchants ?

The Page Meta Tags Rules will simplify the management of meta tags using a rules-based approach. It will maximize search engine rankings and boost ROI by acquiring new merchants organically and improve conversion by targeting shoppers that are actively searching for products.

The Page Meta Tag Rules feature provides:

- Rule creation and content localization
- Rule sharing across catalogs and sites
- Script API methods to integrate the meta tag content into the storefront
- Preview capabilities on category, product, content and folder level
- Capability to import/export meta tags and meta tag rules

Are there any requirements to use Page Meta Tag Rules?

No - There are no requirements needed prior to using the Page Meta Tag Rules.

Do merchants have to pay to use Page Meta Tag Rules?

No - Business Manager users can use the Page Meta Tag Rules free of charge.

Can rule based meta tags be applied and modified on dynamically created pages (such as facet navigation filters)?

Yes - page meta tags can be obtained from API objects Product/Content, Product-/ContentSearchModel or Site. If your dynamic page uses one of these objects you can obtain page meta tags. By using conditions in the rule syntax like IF, THEN, ELSE, OR, CONTAINS, etc. merchants are able to solve complex scenarios for applied filters / facets and even for specific filter values and use this values within the meta content.

Can meta tags be manually edited on a page by page basis if used for dynamically created pages?

Yes - rules follow an inheritance and can be overridden for each product, category, content or folder (following the path - more details: https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/SearchEngineOptimization/MetaTagRules.html). Each rule can be created localized. Rules are executed on request. This means, no index rebuild is required and the rules can potentially use all dynamic information from the page (i.e. applied filters, request url, price information and eve search phrase used by the storefront Merchant).

What meta tags are supported? (i.e. page titles, meta descriptions, meta robots, meta keywords etc)

Business Manager users can create up to 50 page meta tag definitions (following a naming schema). A definition can be used to create page meta tag rules, which can be assigned (depending on the type) to the corresponding objects, for example 'PRODUCT_DETAIL_PAGE' can be assigned to category or product.

All tags with the attribute title (<**title**>), name (<meta **name**="..." content="...">) and property (<meta **property**="..." content="...">) are supported out of the box. In addition to the basic tags like Page Title, Meta Description, Meta Keywords, Site Verification, Meta Robots also Open Graph tags (for Facebook etc.) and Twitter Card tags are supported > see https://github.com/joshbuchea/HEAD for some further useful tags.

Is it required to maintain rules on product level?

No - meta tag rules for products (or content assets) are intended to be assigned to the *root (or any child)* category which will inherit the rule across all (child) categories and their products for site's catalog. However, it is possible to override rules by direct assignment to a product.

Which attributes can Business Manager users incorporate into the rules? Where can merchants see these attributes?

System and custom attributes on the corresponding object, which can be reviewed in the management module under Administration are supported. However, not all attribute definitions are support, e.g. password fields or HTML are not. System and custom objects are maintained at Administration > Site Development > System Object Types / Custom Object Types.

Can Business Manager users reference to image URLs hosted in a 3rd party solution?

Yes - the default image logic will be used to obtain the URL, means if images are defined properly it will work.

Is there an option set meta tag content for specific applied filters / facets?

Yes - the feature uses a rule based approach and rules can contain logical expressions it is possible to configure the Robots Meta Tag for 'specific filter/facet' combinations.

Can merchants create rel=canonical tags?

No - the canonical tag is currently not supported out of the box but can be used with some customization leveraging the Meta Tag Rules Script APIs. Enhancing the feature to support this use case is considered for further releases. There is currently no functionality in place to support this requirement accordingly, however there might be one in future within page meta tag rules (or an other API, depending on requirements for this functionality - need to be defined).

Can merchants create dynamic headings (<H1> ...<H6> tags)?

No - currently only title, name and property are supported meta tag types. This use case might be fulfilled with customization, depending on the meta tag Id and custom code extension. Headings are currently not supported out of the box but can be used with some customization leveraging the Meta Tag Rules Script APIs. Enhancing the feature to support this use case is considered for further releases.

How can merchants restrict the character count in the meta tags?

This can be done in HTML code via markup. This requirement is scheduled for any subsequent update release and is currently not part of the GA release. There might be a limitation within the meta tag definition and/or in the rule syntax.

Where are all of the available operations and expressions listed?

Please see Commerce Cloud B2C Infocenter with examples.

Are rules exposed via Script API?

Only (evaluated rule) content is available at script API level, based on the current page context.

Can merchants combine meta information created on product/category level and the rule based approach?

Yes - by using System Object or Custom Object types allows to check for existing content and combine it within the meta tag rules. Rules can be assigned to different categories/products and are collected, following an inheritance. However, Business Manager users can't inject content, created based on rule A into rule B.

Why can't merchants see all products/content assets in the list picker (product & content asset assignment) in Business Manager?

The product / content asset assignment modal is currently restricted to 20 search hits to provide a great user experience in terms of Business Manager search. If merchants don't see the desired search results in the product or content assignment in Business Manager, they can narrow down the search result list by using a more specific product id in the search bar.

What are "undefined locales" filter in the Page Meta Tag Rules module in Business Manager?

The search for undefined locales will return page meta tag rules that has no rule defined for the selected locales. The search results will display rules which have no localization for the given locales.

Do child category assignments override parent category assignments? Or is there a priority order the user can set?

Yes - rules assigned to child categories are always overriding parent category rule assignments (for the same meta tag). The rule lookup will firstly try to obtain a rule within the current requested locale (following the catalog fallback path).

Can merchants still override the rule on a specific product/category?

Yes - rules can be assigned to each object (category, product, content folder and content asset) and override the rule from the parent object (ie parent category or folder). Products are considered as child objects of (parent) categories. Similar content folder and content assets.

Can the rules contain custom attribute?

Yes - example: ${IF Category.custom.robotsFlag CONTAINS Constant('true') THEN Constant('noindex,nofollow') ELSE Constant('index,follow')}

How can merchants override meta tags on a page by page basis if used for dynamically created pages?

Rules follow an inheritance and can be overridden for each product, category, content or folder (following the path - more details in infocenter). Each rule can be created localized. The homepage (site) meta tag rule can be only localized. Rules are executed on request. This means, no index rebuild is required and the rules can potentially use all dynamic information from the page (i.e. applied filters, request url, price information and eve search phrase used by the storefront user).

Is there a list of all the rules that can be created?

No - a rule can contain static and dynamic attributes (i.e. “Buy ${Product.name} for ${ProductPrice.min} ${ProductPrice.currency} | ${Site.displayName}”). Since the combination options are infinite, we can't provide a list of *all* the rule. However, in addition to the System & Custom Object overview in Business Manager, Commerce Cloud Infocenter (see https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/SearchEngineOptimization/MetaTagRuleSyntax.html) provides a list of dynamic attribute which can be used within the rules as well as some examples. In near term we are going to provide a list with some best practice rules as well.

Is Commerce Cloud planning to support Page Meta Tag rule on Pipelines pages?

The feature supports Controller and Pipeline based Storefront development models. In terms of page types, the currently supported page types are Product Detail Pages, Product Listing Pages (Category pages), Search Result Pages, Home Page, Content Listing Pages (Library) and Content Detail Pages (Content Assets). “Pipeline Pages" like Order-History, Checkout Pages, Error Pages,... (basically pages/URLs listed at Merchant Tools > SEO > URL Rules > Pipeline URLs) are currently not supported out of the box.

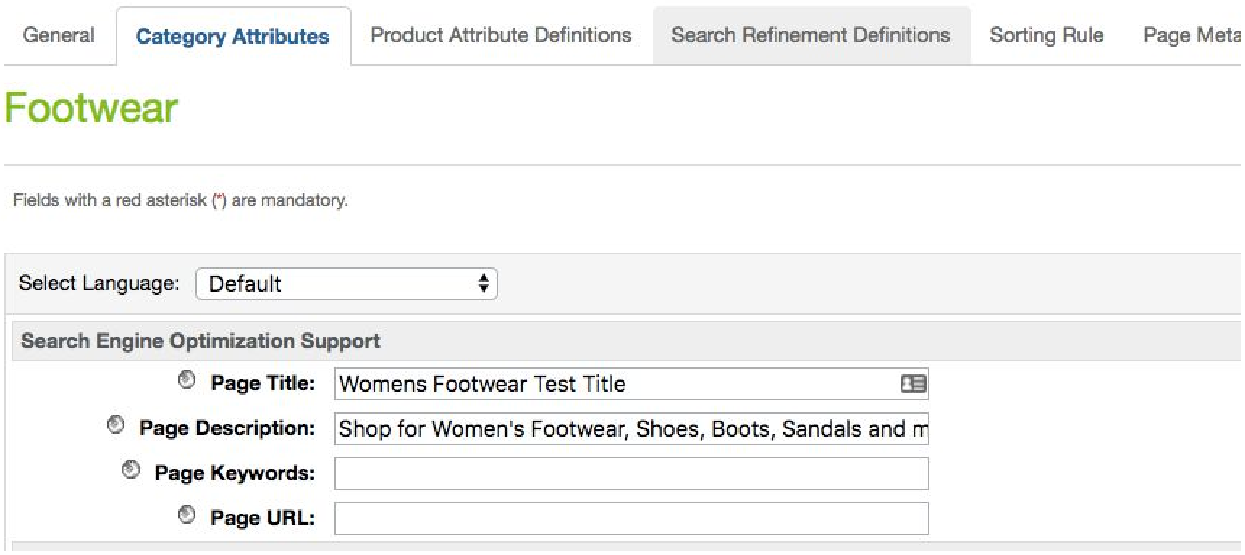
Is controller / pipeline page meta data included in this feature?

No - only the objects in the documentation are included (site, product, category, etc) in respective of the page scope of course. Including Page Meta Tag Rules into pipeline / controller pages is on the roadmap. However, note that this feature can be used in Pipeline or Controller based implementations.

Can merchants combine meta information created on product/category level and the rule based approach?

Yes - whether existing meta tags will be overridden depends on the storefront integration of the code (please see documentation at https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/SearchEngineOptimization/MetaTagRulesSiteIntegration.html). In general, the rule syntax is very powerful and merchants can combine existing (i.e. in BM manual created and optimized) meta data with rule generated data by using the meta tag rule feature only. By adding some minor logic to the rules, a rule can check for existing content maintained in BM and if available, show the existing content. If no content is available, use the rule generated content.

Example: merchandiser maintains "Page Title" on some categories and would like to show this page title in the storefront:



Footwear

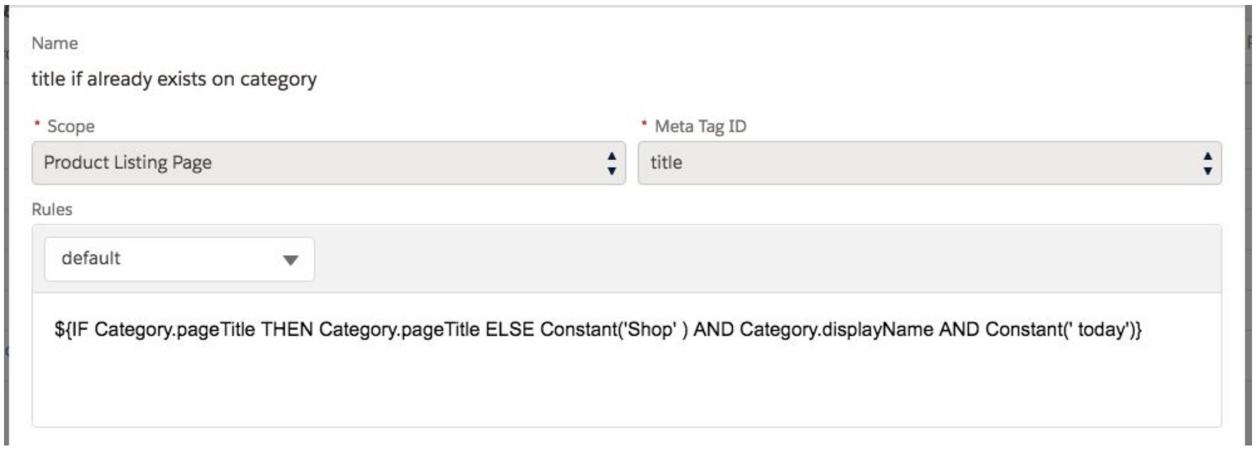
Fields with a red asterisk (*) are mandatory.

Select Language: Default

Search Engine Optimization Support

Page Title:	Womens Footwear Test Title
Page Description:	Shop for Women's Footwear, Shoes, Boots, Sandals and m
Page Keywords:	
Page URL:	

Even by fully implementing the new approach, merchandisers can reference this content in the rules and reuse this existing content with the following simple rule:



Name
title if already exists on category

* Scope: Product Listing Page
* Meta Tag ID: title

Rules
default

```
${IF Category.pageTitle THEN Category.pageTitle ELSE Constant('Shop' ) AND Category.displayName AND Constant(' today')}
```

*If there is a category.PageTitle populated then show it, else show this dynamic rule of Category Display name inserted into text of ‘Shop Footwear today!’

For Products, use a similar syntax: ${IF Product.pageTitle THEN Product.pageTitle ELSE Constant('Buy') AND Product.name AND Constant('today!')}

The clear advantage of this approach is:

- Merchandiser can reuse existing content and use the rule based content as a fallback (i.e. page title is missing)
- Rules have a build-in inheritance mode, i.e. assigning the above mentioned rule to root in storefront catalog generates content for all products in this catalog
- Adding new meta tags to the storefront doesn't require additional development effort any more > if merchandisers have adopted the meta tag rules feature and want to add for example open graph tags for facebook, it can be now accomplished within minutes and not days or even weeks (time saver)

Merchants can keep their current implementation and serve some meta data from categories, products or directly from the templates and some meta data with the rule based approach.

Will the new assigned meta tag title also appear in the Search Engine section of the Catalog and Product set-up pages?

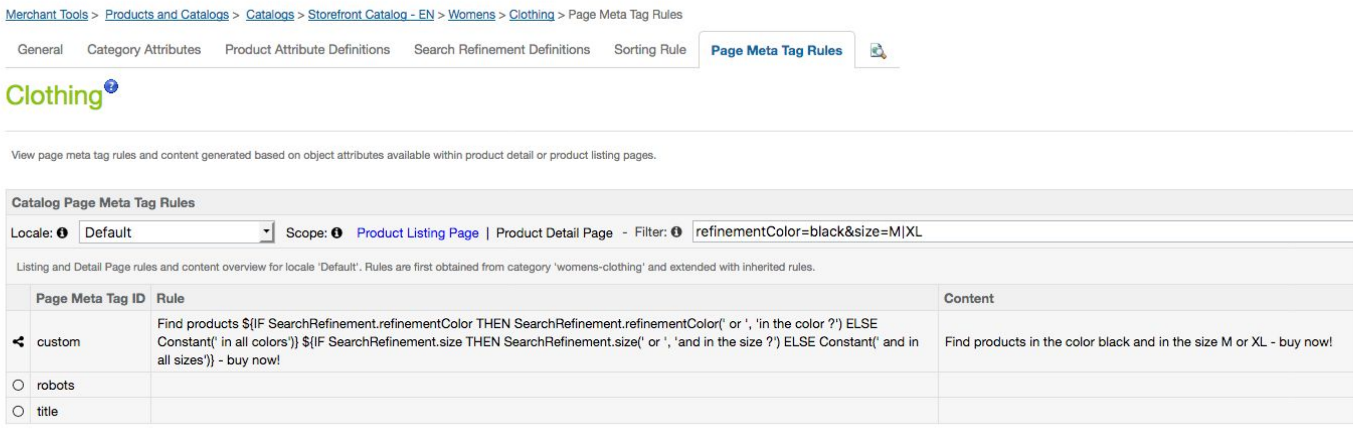
No - the section *Search Engine Optimization Support* in the *General* tab of an object contains the merchant imported / maintained data only and the rules won't override the data. However, a preview is available on each object (i.e. product, category, folder or asset) in the new *Page Meta Tag Rules* tab.
Important note: depending on the storefront integration, merchants can still leverage the existing SEO meta data maintained at *Search Engine Optimization Support*, can use new rule based approach or can combine both approaches by creating a reference in a rule to the existing data:

- Example rule: ${IF Product.pageTitle THEN Product.pageTitle ELSE Product.name}
- Explanation: this rules uses data maintained in object *pageTitle* (if available) in title tag, else (i.e. if missing) it uses the product name in title tag

How can I preview meta tag content for refined categories or search pages?

On the category or folder object you can trigger dynamic elements like filters/refinements or even search phrases by adding these parameters to the preview text field in the “Page Meta Tag Rules” tab:

- Example rule: Find products ${IF SearchRefinement.refinementColor THEN SearchRefinement.refinementColor(' or ', 'in the color ?') ELSE Constant(' in all colors')} ${IF SearchRefinement.size THEN SearchRefinement.size(' or ', 'and in the size ?') ELSE Constant(' and in all sizes')} - buy now!
- Preview parameter: refinementColor=black&size=M|XL


Merchant Tools > Products and Catalogs > Catalogs > Storefront Catalog - EN > Womens > Clothing > Page Meta Tag Rules
General
Category Attributes
Product Attribute Definitions
Search Refinement Definitions
Sorting Rule
Page Meta Tag Rules

Clothing

View page meta tag rules and content generated based on object attributes available within product detail or product listing pages.

Catalog Page Meta Tag Rules

Locale: Default   Scope: Product Listing Page | Product Detail Page - Filter: refinementColor=black&size=M|XL

Listing and Detail Page rules and content overview for locale 'Default'. Rules are first obtained from category 'womens-clothing' and extended with inherited rules.

	Page Meta Tag ID	Rule	Content
	custom	Find products ${IF SearchRefinement.refinementColor THEN SearchRefinement.refinementColor(' or ', 'in the color ?') ELSE Constant(' in all colors')} ${IF SearchRefinement.size THEN SearchRefinement.size(' or ', 'and in the size ?') ELSE Constant(' and in all sizes')} - buy now!	Find products in the color black and in the size M or XL - buy now!
○	robots		
○	title		

Is the quota rule for meta tag maximum global for an instance, or per page ?

Yes - the quota limit for Meta Tag Rules [2000 soft limit (warning) and 3000 hard limit (error)] is applied to all sites within one instance. Furthermore, max. 50 different meta tags can be created per instance which limits the possible meta tags per page up to 50 as well. However, pages typically contain between 10-30 relevant (dynamic) meta tags max.

What if the underlying data is not as accurate as required by the dynamic meta tag, i.e. the required ${Product.typeDescription} is missing? How would it be rendered, as null?

If the object is not created within the instance (ie "Product.typeDescription" is not defined in Administration) Business Manager displays a validation error.Validation error is only thrown in case of rule engine syntax is violated. In case of the incorrect attribute is included, the dynamic part simply result in an empty string or if the whole rule is empty it will not returned from the scripting API. E.g. the rule 'The product ${Product.unknown}.' will return 'The product .'.

If the object is available but doesn't contain data for the assigned page, nothing will be rendered for this object (i.e. the tag won't show up in HTML code).

What is the difference between content listing / content detail page?

"Content Listing Pages" are "Content Library"/"Content Folder" objects. "Content Detail Page" are "Content Assets". Since content folder and content asset are Commerce Cloud specific terms and not industry terms, but the feature is/can be used by SEO agencies and in long term by other Salesforce clouds, the decision has been made to use the industry terms.

Rules are shared across sites - How does it work if the merchant wants different rules on different sites?

Rule sharing works like assigning products to different catalogs. Means rule A can be assigned to Site A & Site B or to Site A only or to Site B only. In the latter case merchants can create two rule, one for Site A and one for Site B.

Can rule content be specific and conditional to attributes of the categories/products(IDs)?

Yes -example: ${IF Category.ID CONTAINS Constant('womens') THEN Constant('Get your ') AND Category.ID AND Constant(' here gurl!!') ELSE Constant('Find the best quality and prices of ') AND Category.ID AND Constant(' here!')} | ${Site.displayName}

Which approach should merchants use to prevent crawling or indexation of URLs?

In general, there are multiple approaches merchants could use to prevent crawling or indexation of pages:

- Block URLs in the *Robots.txt file* (https://moz.com/learn/seo/robotstxt)
- Set "Noindex" directive with the *Robots Meta Tag* (https://moz.com/learn/seo/robots-meta-directives)
- Use *Canonical Tags* (aka "rel canonical") (https://moz.com/learn/seo/canonicalization)

Important point to consider: the SEO strategy should make sure merchants are implementing/using a consistent and search engine conform behaviour (https://support.google.com/webmasters/answer/93710?hl=en).

Roll-out and Timeline Schedule

When will Page Meta Tag Rules be Generally Available?

The Page Meta Tag Rules is planned to be Generally Available by 18.8 Digital Release Deployment.

What is the Page Meta Tag Rules deployment schedule?

Here's the planned deployment schedule for Page Meta Tag Rules:

SIGs
July 31, 2018

PIGs
Phase 1: August 7, 2018
Phase 2: August 14, 2018

On-boarding merchants and Activation

How can Commerce Cloud merchants enable Page Meta Tag Rules?

The Page Meta Tag Rule UI and Page Meta Tag UI will be automatically available in Business Manager with Commerce Cloud 18.8 Digital Deployment.

Merchants can manage Meta Tag definitions in *Merchant Tools > SEO > Page Meta Tag Rules > Meta Tag Definitions*

The Page Meta Tag Rules is located in *Merchant Tools > SEO > Page Meta Tag Rules*

Will there be any development necessary (addition of a snippet in the ISML for example) to use the Page Meta Tag feature?

Yes - to make the feature work on the Storefront some code changes need to be made to the platform generated markup -- as well as to the meta-processing logic that is used to generate the tag content. Implementing the Meta Tag features on each of our reference architectures require different execution steps. To simplify this, both sets of instructions are included below.

Note: In order to use the embedded hyperlinks in this section to view the source-code changes outlined below, you will need access to the SiteGenesis and Storefront Reference Architecture codebase repositories in github. If you do not have access to github, you can obtain access by following the instructions found in the Github High Impact Product Brief available via Chatter. To obtain access to these repositories:

- Create an account on bitbucket using your Salesforce E-Mail Address
- Please connect with Jim Lynch (jim.lynch@salesforce.com), provide him with your account information, and request access to both codebase repositories

The embedded hyperlinks are better representations of the code changes necessary to enable this feature -- as they have already been applied to SiteGenesis and the Storefront Reference Architecture.

Implementation Instructions for SiteGenesis

The SiteGenesis code changes for this enhancement can be inspected via the git commit f74e591bbc in bitbucket. Implementing the Page Meta Tag feature for SiteGenesis requires three distinct steps:

1. Update the meta.js script to include logic leveraging the new meta-data based digital APIs to add the rule-based meta tags to its model:

```
// Update the Page Meta tags that are generated based on rules
updatePageMetaTags: function (object) {

        // Check if object wrapped in AbstractModel and get system object
        if ('object' in object) { object = object.object;

        // Only generate page meta-data for the following object classes
        if (object.class === dw.content.Content
             || object.class === dw.catalog.Product
             || object.class === dw.catalog.ProductSearchModel
             || object.class === dw.content.ContentSearchModel) {

                // Retrieve the meta-data object from the request
                var pageMetaData = request.pageMetaData;

                // Add the page meta-data tags for this object
                pageMetaData.addPageMetaTags(object.pageMetaTags);

        }

}
```

2. Embed references to the meta.js file in the Page.js, Product.js, and Search.js controllers -- and add logic to update the meta-data content and page-tags configured in the previous setup.

   These code entries take the current platform object (ex. a content asset, a specific product, or a collection of search results) being requests by the user and retrieves the raw object data (ex. the content asset name, product name, or category name) necessary to render page meta-tags in accordance with the rule definition created via Business Manager.

Common Entry for All Impacted Controllers

The instantiation entry is common for all scenarios and should be applied to the Page.js, Product.js, and Search.js controllers.

```
// Instantiate the meta script
meta = require('~/cartridge/scripts/meta');
```

Content Page Meta-Tag Content

This change should only be applied to the Pages.js controller -- as it is used to seed the meta-tag content for content pages.

```
// Update the page meta-tag content for a given content page
meta.update(content);
meta.updatePageMetaTags(content);
```

Product (PDP) Page Meta-Tag Content

This change should only be applied to the Product.js controller -- as it is used to seed the page-specific meta-tag content for the PDP.

```
// Update the page meta-tag content for a given product
meta.update(product);
meta.updatePageMetaTags(product);
```

Product-Search (PLP) Page Meta-Tag Content

This change should be applied to the product-search section of the Search.js controller -- as it is used to seed the page meta-tag content for the product category pages.

```
// Update the page meta-tag content as part of a given product category search
if (productSearchModel.category) {
     meta.update(productSearchModel.category);
}

// Update the page meta-tag content using the product search results
meta.updatePageMetaTags(productSearchModel);
```

Content-Search Page Meta-Tag Content

This change should be applied to the content-search section of the Search.js controller -- as it is used to seed the page meta-tag content for content searches (ex. searching content assets).

```
// Update the page meta-tag content as part of a given content search
if (contentSearchModel.folder) {
     meta.update(contentSearchModel.folder);
}

// Update the page meta-tag content using the content search results
meta.updatePageMetaTags(contentSearchModel);
```

3. Extend the htmlhead.isml template to include logic to render the page meta-tags based on the rule configurations for a given storefront page.

Rendering the Page's Title Meta-Tag

This snippet demonstrates how to render the page's title meta-tag.

```
<iscomment>Set the page title -- if there is no rule based content available</iscomment>
<isif condition="${!pdict.CurrentPageMetaData.isPageMetaTagSet('title')}">

     <iscomment>Is this a production system?</iscomment>
     <isif condition="${dw.system.System.getInstanceType() != dw.system.System.PRODUCTION_SYSTEM}">

          <iscomment>If not, then render the title -- plus the site / revision info</iscomment>
          <title>
```

```
                ${pdict.CurrentPageMetaData.title} |
                ${Resource.msg('global.site.name', 'locale', null)} |
                ${Resource.msg('revisioninfo.revisionnumber', 'revisioninfo', null)}
            </title>

     <iselse/>
            <iscomment>Otherwise, just render the tile (this is for production)</iscomment>
            <title>
                <isprint value="${pdict.CurrentPageMetaData.title}" encoding="off"/>
            </title>

     </isif>
</isif>
```

Rendering the Page's Description Meta-Tag

This snippet demonstrates how to render the page's description meta-tag.

```
<iscomment>Set the page-description meta-tag if a rule has been configured</iscomment>
<isif condition="${!pdict.CurrentPageMetaData.isPageMetaTagSet('description')}">

     <iscomment>Open the meta-tag</iscomment>
     <meta name="description" content="

     <iscomment>Only render the description content if it's configured for this page</iscomment>
     <isif condition="${!empty(pdict.CurrentPageMetaData.description)}">
          ${pdict.CurrentPageMetaData.description} |
     </isif>

     <iscomment>Append to storeName constant to the description</iscomment>
     ${Resource.msg('global.storename','locale',null)}

     <iscomment>Close the description meta-tag</iscomment>
     "/>

</isif>
```

Rendering the Page's Keywords Meta-Tag

This snippet demonstrates how to render the page's keywords meta-tag.

```
<iscomment>Set the page-keywords meta-tag if a rule has been configured</iscomment>
<isif condition="${!pdict.CurrentPageMetaData.isPageMetaTagSet('keywords')}">

     <iscomment>Open the meta-tag</iscomment>
     <meta name="keywords" content="

     <iscomment>Only render the keyword content if it's configured for this page</iscomment>
     <isif condition="${!empty(pdict.CurrentPageMetaData.keywords)}">
          ${pdict.CurrentPageMetaData.keywords}
     </isif>

     <iscomment>Append to storeName constant to the keywords entry</iscomment>
     ${Resource.msg('global.storename','locale',null)}

     <iscomment>Close the description meta-tag</iscomment>
     "/>

</isif>
```

Rendering the Collection of Rule-Driven Meta-Tags

This snippet demonstrates how to render the full collection of configured meta-tag rules for a given storefront page. Please note that new types (other than name and property) may be introduced in the future.

```
<iscomment>Render rule based page meta tags (title, name, or property)</iscomment>
<isloop items="${pdict.CurrentPageMetaData.getPageMetaTags()}" var="pageMetaTag">

     <iscomment>Is there a title tag to render?</iscomment>
     <isif condition="${pageMetaTag.title}">

           <iscomment>If so, then render the page title</iscomment>
           <title><isprint value="${pageMetaTag.content}"/></title>

     <iscomment>Otherwise, check if the tag to render is a 'name' tag</iscomment>
     <iselseif condition="${pageMetaTag.name}">
```

```
          <iscomment>If so, render the tag's content following the name convent[illegible]/iscomment>
          <meta name="<isprint value="${pageMetaTag.ID}">"
                content="<isprint value="${pageMetaTag.content}">
          ">

     <iscomment>Lastly, check if the tag to render is a 'property' tag</iscomment>
     <iselseif condition="${pageMetaTag.property}">

          <iscomment>If so, render the tag's content following the property convention</iscomment>
          <meta property="<isprint value="${pageMetaTag.ID}">"
                content="<isprint value="${pageMetaTag.content}">
          ">

     </isif>

</isloop>
```

Implementation Instructions for Storefront Reference Architecture

To update an existing SFRA storefront with logic to render rule-driven Page Meta Tags, please execute the following steps:

1. Visit the Storefront Reference Architecture code repository on bitbucket.
2. Navigate to the 794af38 commit -- which contains the Page Meta Tag changes.
3. Manually merge in these changes to your app_storefront_base, modules, and test directories. This can be done through a manual copy / paste of each file, or through a cherry-pick of the commit via git.

Note: No actual storefront code-changes should be required to implement Page Meta Tags on storefronts leveraging the latest Storefront Reference Architecture. This is an example of one of the many differentiating benefits that come with adopting the Storefront Reference Architecture.

What is the syntax that merchants need to use for creating Meta Tag Rules?

Page meta-tag rules contain both static and dynamic elements. Static elements (ex. straight text) are printed directly as-is, while dynamic elements must be defined within the platform's scripting evaluation syntax ${dynamic-expression-to-be-evaluated}.

Dynamic Text Generation Example

In the example below, a rule is configured promoting a product name and it's category -- and appends the site's display-name at the end of the tag content (dynamic elements are highlighted in black). The end result is a dynamic text string that can be used as a page-title for a given product.

```
Find ${Product.name} in ${Category.displayName} today | ${Site.displayName}
```

Example of the Rendered Result in the Storefront

When rendered in the storefront, the dynamic expressions are replaced with the object properties embedded in the rule definition.

Here, the product name is taken from the current page's Product. Similarly, the current category's display name is taken from the active Category. To complete the tag, the Site's merchants-facing display-name is appended (rendered output of dynamic elements are highlighted in black).

```
Find Apple iPod Classic in iPod & MP3 Players today | SiteGenesis
```

Page Meta-Tag Rules Object Access

During the creation of meta-tag rules, the rules syntax includes access to the following objects and their property values. The properties of these objects can be referenced within Meta Tag Rules via the dynamic evaluation syntax:

- Product
- Category
- Folder
- Content
- Site

For details on the object properties available and their represented values -- please use the links above to access the class definitions for these objects via the platform documentation. While defined system and custom values are available within these objects -- there are properties that cannot be used: specifically password properties as well as other unsupported values. Consult the platform release documentation for additional details on specific unsupported values.

Convenience Objects

Some storefront information (such as product price or product image url) can only be made available through convenience objects. These are objects that exist solely within the context of meta-tag rule creation -- and are designed to easily expose data that usually requires complex access-logic via the Digital API.

ProductImageURL

The ProductImageURL object can be used to access the image properties for a given product. The ProductImageURL object can be accessed by specifying the viewType of the image url to be retrieved. Specifying a view-type will return the static imageUrl for that product's viewtype.

Property	Description and Return Result
viewtype	Represents the view-type for a given product; returns the static product image url that maps to the specified view-type

In the example below, the 'medium' view-type product image url is rendered via a dynamic expression.

Dynamic Expression

```
${ProductImageURL.medium}
```

Rendered Result

```
https://www.site-genesis.com/on/company.static/-/Sites-electronics-catalog/default/
dw32ee7837/images/medium/ipod-classic-silver.jpg
```

ProductPrice

The ProductPrice object can be used to access price and currency information for a given product.

Property	Description and Return Result
min	Represents the minimum price for a given product
max	Represents the maximum price for a given product
currency	Represents the merchants-facing currency code for a given product based on the current user's locale and session

In the example below, the price-range and currency are rendered via a dynamic expression.

Dynamic Expression

```
${ProductPrice.min} - ${ProductPrice.max} ${ProductPrice.currency}
```

Rendered Result

```
15.99 - 25.99 USD
```

SearchRefinement

The SearchRefinement convenience object can be used to access all selected search-refinements selected by a merchants as part of their search.

Property	Description and Return Result
refinementId	Returns the individual merchants-facing selected refinement values
refinementId(delimiter)	Specifies the delimiter used to join multiple refinement values.
refinementId(delimiter, phrase)	Specifics the delimiter used to join multiple refinement values -- and the phrase used to highlight the selected values. Uses a ? as a replacement placeholder within the phrase.

Dynamic Expression

```
${SearchRefinement.refinementColor(' and ', 'for the color ?')}
```

Rendered Result

Generates the content *' '* for the search page without any selection. If the color ***black*** is selected, the content `'for the color black'` is generated. If ***blue*** is later selected, the content `'for the color black and blue'` is generated.

SearchPhrase

The SearchPhrase convenience object can be used to access the search phrase specified by a user as part of an existing storefront search.

Property	Description and Return Result
SearchPhrase(phrase)	Returns the search-phrase specified by the user as part of an existing storefront search withinthe phrase provided to the convenience object.

Dynamic Expression

```
${SearchPhrase('for the search phrase ?')}
```

Rendered Result

Generates empty content for the search page without a search phrase. If the search phrase ***shoes*** is added, the content `'for the search phrase shoes'` is generated.

OriginalRequestURL

The OriginalRequestURL convenience object can be used to access the current page url used to access the storefront by a given merchants. The url is printed exactly as it was requested via the browser.

Dynamic Expression

```
${OriginalRequestURL}
```

Rendered Result

```
http://www.sitegenesis.com/mens/
```

Logical Operators

The dynamic evaluation syntax also supports the use of logical operators which can be leveraged to construct complex rules designed to conditionally render tag content. These operators include:

- **OR**
- **AND**
- **ELSE**
- **IF-THEN**
- **EQ**
- **CONTAINS**

The following examples demonstrate how logical operators can be used to dynamically change meta-tag rule content based on the conditionality of the elements within a given rule.

OR Expression

The logical OR expression means that either the left or right side of a dynamic expression must evaluate as true -- and supports dynamically joining both elements via the expression argument.

The following script:

```
Find ${Product.name OR(' in ') Category.displayName} today | ${Site.displayName}
```

Generates this content by default:

```
Find Apple iPod Classic in iPod & MP3 Players today | SiteGenesis
```

When the category **displayName** is removed -- the following content is generated:

```
Find Apple iPod Classic today | SiteGenesis
```

AND Expression

The logical AND expression means that both left and right side must evaluate true -- and supports dynamically joining both elements via the expression argument.

The following script:

```
Find ${Product.name AND(' in ') Category.displayName} today | ${Site.displayName}
```

Generates this content by default:

```
Find Apple iPod Classic in iPod & MP3 Players today | SiteGenesis
```

When the category **displayName** is removed -- the following content is generated:

```
Find today | SiteGenesis
```

ELSE Expression

The logical ELSE expression creates a fallback position for two elements; if the left side evaluates false, evaluate the right side.

The following script:

```
Find ${Product.name AND(' in ') Category.displayName ELSE Product.ID AND(' in ')
Category.ID} today | ${Site.displayName}
```

Generates this content by default:

```
Find Apple iPod Classic in iPod & MP3 Players today | SiteGenesis
```

When the category **displayName** is removed -- the following content is generated (note how the ELSE condition renders the product / category identifiers instead of the names):

```
Find apple-ipod-classic in electronics-digital-media-players today | SiteGenesis
```

IF-THEN Expression

The logical IF-THEN expression means that if the left portion of the expression evaluates as true -- then evaluate the right portion of the expression when rendering meta-tag content.

```
IF leftExpression THEN rightExpression
```

The following script:

```
${IF SearchRefinement.refinementColor OR SearchPhrase THEN
Constant('noindex,nofollow') ELSE Constant('index,follow')}
```

Generates this default content:

```
noindex,nofollow
```

Please note that this content is generated when a refinement color or searchphrase exists. If color or search phrase is found, the content 'index,follow' is generated.

ENDIF Expression

IF-THEN-ELSE expressions can now be nested. In these scenarios, the ENDIF expression should be placed at the end of the conditional statement -- acting as a terminator for the IF-THEN-ELSE logic.

```
IF expression THEN
     IF otherExpression THEN [...]
          ELSE [...]
          ELSE [...]
     ENDIF
     ELSE [...]
ENDIF
```

EQ Expression

The EQ expression means that the left and right side of the expression must be equal -- and when this occurs, the expression will return a value of true. Otherwise, it returns a value of false.

The following script:

```
${IF ProductPrice.min EQ ProductPrice.max THEN ProductPrice.min ELSE
ProductPrice.min AND('-') ProductPrice.max} ${ProductPrice.currency}
```

Generates this default content:

```
199.00 USD
```

Please note that the content `'199.00 USD'` is generated for the product `'apple-ipod-classic'` which has a single price value. For a product with a min price '90' and max price '120', the content `'90.00-120.00 USD'` is generated.

Contains Operator

The CONTAINS operator allows meta-tag elements to be inspected for a specific string. If the string is found in the source element, the operator returns a value of true. Otherwise, a value of false is returned by the operator. This value can then be used to dynamically drive the rendering of tag content.

The following script:

```
IF SearchPhrase CONTAINS Constant('shoes') THEN ‘With Shoes’ ELSE ‘Without’
```

Generates this default content:

```
‘Without’
```

And Generates this content when the SearchPhrase contains the word shoes:

```
‘With Shoes’
```

Please consult the platform documentation for additional details on these meta-tag script expressions -- as well as examples for how to collectively leverages these expressions to dynamically generate precise meta-tag content.

When I use the object *SearchPhrase* in the rules together with Search Autocorrection - which search phrase is used?

Search phrases can be included in meta tag content. Currently, the original search phrase is used only, not the corrected/autocompleted one.

How to use the CONTAINS operator?

The CONTAINS operator can be used to check for specific string values of an object. Example: “${IF SearchPhrase CONTAINS Constant('shoes') THEN ... ELSE ...}”

Do I assign my product page meta tag rules to the master catalog or the storefront catalog?

To the Storefront catalog.

Are rules exposed via Script API?

Only (evaluated rule) content is available at script API level, based on the current page context.

Are special characters encoded in the Storefront?

The script API will return unencoded content, however e.g. isprint can be used to print this content html encoded.

Once the script is added to the template, does it still respect prior template driven syntax if no rules are defined for that object?

Custom code can be created to fallback to native object description if not available within page meta tag rules (e.g. null check on dw.catalog.Product._getPageMetaTag_), however the page meta tag rule can fallback to the native value, which seem to be the better approach (e.g. ${Product.description}).

Can the feature used with controller & pipeline based development model?

Yes, this feature can be used in both - Pipeline or Controller based - implementation models.

How can merchants deploy the Page Meta Tag Rules and Definitions to their Production site(s)?

Merchandisers will need to replicate the Page Tag rules and Definition to their Production instance. This only include the rules and definitions. Assignments are in the corresponding container - e.g. Product/Category assignments within Catalog replication group.

What is the replication flow for code and data?

Firstly the code, which should break nothing without existing rules - if implemented correctly. Secondly page meta tag definition and rules can be replicated within a new replication group 'Page Meta Tags'. Assignments to these rules are replicated within Catalog (Categories/Products), Library (Folder/Content) and Repository (Site) and can be replicated subsequently or parallel to definition/rule. The feature is designed to not break if replicated in the wrong order, however obviously rules might be not evaluated if only assignments are replicated.

How can I combine the feature with existing rules (manually entered in BM or hard-coded)?

Manual / imported meta data: a meta tag rule can reference to existing (imported meta data) and override this data OR generate data only if imported data is missing (fallback strategy). Hard coded: you needs to adjust storefront code OR use the feature for other (non hard-coded) meta tags to avoid duplicate meta information.

Will merchants be able to test Page Meta Tag Rules on their sandboxes?

Yes, Merchants can test the Page Meta Tag Rules on their Sandboxes. The process will be the same as if they are creating meta tag definitions and tag rule on their Staging instance.

Troubleshooting and Escalation Process

What are the anticipated issues that support should look out for with Page Meta Tag Rules?

Below is the list of anticipated issues that support might encounter:

1. **Why does the meta tag content not change after a filter has been applied?**
   There are a couple of reasons why Page Meta Tag Rules are not updated in the Storefront:
   - Verify the correct rule syntax
   - Verify the correct rule assignment (i.e. to the desired storefront catalog)

- Ajax is not supported