Sitecore Experience Commerce 10.0 – Release Notes

August 14, 2020 – Released Sitecore Experience Commerce 10.0

October 15, 2021 – Released an updated XC host file as described in KB1000801, for connecting to Identity Server. This is required to make XC 10.0 compatible with XP 10.0 Update-3.

Contents:

New Features and Improvements

Dynamic Bundles

Area	Features / Improvements – Dynamic Bundles	ID
The Bundles feature represents a collection of sellable items that are grouped together to be managed and sold as a distinct product. In this release, the existing Static Bundles feature has been extended to provide more flexibility in the products that make up the bundle. The merchandiser can now configure and the customer can now select from optional and upgradeable products within the bundle. For example: Basic bundle: a tablet device bundled with a power adapter Optional: add a screen protector or headphones to the bundle Upgradeable: Select from a device with 16GB, 32GB, or 64GB of memory		
Storefront	Added a new default Storefront Bundle component rendering variant that extends the existing Static Bundles feature. The customer can select from optional and upgradeable products offered within the bundle. The customer may change the quantity of bundled items by configuration. The new Dynamic Bundle override template is configured on the catalog for new sites. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/commerce-catalog-renderings.html and https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-product-bundle-rendering.html	277510
Business Tools	Added the ability for the merchandiser to create a new Dynamic Bundle from a Catalog or Category by adding it to the list of Sellable Items using the Add Bundle action. The merchandiser can optionally set a default price at the bundle level (e.g. normally lower than the sum of default constituent items). List price and price card pricing continues to apply to bundles, like in previous releases. The bundle price then gets recalculated from this default when the customer selects options, upgrades, and quantities. Alternatively, the bundle price can be based on the sum of its constituent items (e.g. with no bundle level discount). The	287658

Area	Features / Improvements – Dynamic Bundles	ID
	merchandiser also specifies the quantity range for optional items within the bundle. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/dynamic-bundles.html	

Promotion – Free Gift with Purchase

Area	Features / Improvements – Free Gift with Purchase	ID
A new Free Gift with Purchase promotion benefit type has been added, allowing the merchandiser to configure one or more free gifts that a customer is awarded when purchasing a product. The gift(s) can be automatically added to the cart, or requiring the customer to select the gift(s) or select from gift options or variants.		
Storefront	Added a new Free Gift Selection component (FGS) that presents the customer with a choice of free gifts based on one or more applicable free gift promotions. The customer can optionally claim one or more free gifts according to configured promotions. The FGS component supports communication with other components on the same page to reflect claimed free gifts or free gifts removed from cart through the new messaging service. The component is implemented based on a Scriban template and the component can be used on both the product detail page as well as the Shopping Cart page. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-free-gift-selection-rendering.html	375557 402828 394440 394396 407397
Storefront	Added the Free Gift with Purchase feature to the Minicart component, making free gifts immediately visible to customers when hovering about the Minicart component to see the cart content. The component also uses the new messaging service to notify other components on the page that a cart line has been deleted, which is now also consumed by the Free Gift Selection component. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/renderings--scriban-.html	407390 394444
Storefront	Added the Free Gift with Purchase feature to the Shopping Cart Lines component, making free gifts immediately visible to customers. The component also uses the new messaging service to notify other components on the page that a cart line has been deleted, which is now also consumed by the Free Gift Selection component. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-shopping-cart-lines-rendering.html	394405
Storefront	Added the Free Gift with Purchase feature to the Order Lines component, making free gifts immediately visible to customers. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-order-lines-rendering.html	406947

Area	Features / Improvements – Free Gift with Purchase	ID
Business Tools	Added the ability for the merchandiser to define the free gift at the sellable item or variant level. They can define maximum quantity that the customer can select among free gift options. The free gift items can be defined as automatically or manually add to cart. The free gift item is required to be in stock. The original price and stock info of the free gift items is returned with API. The cart total does not change with the addition of a free gift. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-promotions.html	374256

Promotions Prioritization

Features / Improvements – Promotions Prioritization	ID
Added the ability for the merchandiser working in the Business Tools to control the sequence of calculating promotions, where multiple promotions apply to the cart. This improves control in how overlapping promotions are intended to apply, which allows better control of how the overall benefits apply to customer and brand. A default priority is set equally for each promotion, then the merchandiser can override the setting with a numerical value. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-promotions.html This feature goes together with the simplification provided by merging the calculation of line and cart level promotions (see 373474 later in this document).	385560

Return Available Promotions for Sellable Items

Features / Improvements – Return Available Promotions for Sellable Items	ID
Added the ability via the Engine API to return approved promotions that qualify for a given sellable item. This adds the ability to return actual pricing that includes the discounted price. It enables discount information to be displayed on the Storefront Product List Page and Product Details Page. Search is used as the mechanism for returning the promotions information such as cart display text, benefit etc. This feature is implemented in the Engine and Commerce Engine Connect for this release; it is not implemented in the Storefront. This feature is turned off by default. It can be turned on via `GlobalCatalogPolicy` and controlled at the individual call level via PolicyKeys: `IgnorePromotions`. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/policy-keys.html and https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/retrieving-available-promotions.html	372054

Deployment via Containers

Features / Improvements – Deployment via Containers	ID
Added a new deployment option based on Docker and Kubernetes. Added a Sitecore Experience Commerce Software Development Kit (SDK) with example scripts to you help build your production container images. A set of prepared sample images is also provided via the Sitecore public container registry, for preparing a sample Commerce solution deployment. Includes containers-based deployment for production (e.g. XP1 based), and for a non-production workstation (e.g. XP0 based). For more information, see two XC Installation Guides for Containers on the Sitecore XC 10 download page: https://dev.sitecore.net/Downloads/Sitecore_Commerce/100/Sitecore_Experience_Commerce_100.aspx	367305

Sample Content Hub Connector

Area	Features / Improvements – Sample Content Hub Connector	ID
Added a sample Engine plugin, with support in Storefront and Business Tools. The connector provides a one way synchronization of product attributes from Content Hub's PCM module and product images from Content Hub's DAM module to the Commerce Engine. The sample was based on Content Hub 3.2, using the Professional Service's "ProSaaS" template for the schema. The feature provides a ready-to-go connector for Content Hub 3.2, and can serve as a base to be extended to serve later Content Hub releases. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/walkthroughs-and-sample-scenarios.html		
Storefront	Added support for site wide configuration of the source for catalog images, which can be either the Content Hub DAM or the Media Library. The default is the Media Library for backward compatibility. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/configure-the-source-for-catalog-images.html	400901
Storefront	Added support for the use of the DAM with the Commerce Search Results component and component level configuration of the source for catalog images. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/commerce-catalog-renderings.html	400901
Storefront	Added support for use of the DAM with the Product Images component and component level configuration of the source for catalog images. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/configure-the-source-for-catalog-images.html	400900
Storefront	Added support for use of the DAM with the Free Gift Selection component and component level configuration of the source for catalog images. For more information, see:	400901

Area	Features / Improvements – Sample Content Hub Connector	ID
	https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/configure-the-source-for-catalog-images.html	
Storefront	Added support for use of the DAM with the Order Lines component and component level configuration of the source for catalog images. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/configure-the-source-for-catalog-images.html	400901
Storefront	Added a new embedded Scriban function `sc_imagelink` that returns the URL for the main image of the specified catalog item based on the configured catalog images source. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/the-embedded-commerce-functions-for-scriban-templates.html	416706
Business Tools	Added the ability for the Business Tools to automatically display the data that has been synchronized from the Content Hub's PCM and DAM modules to the XC catalog. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/sample-sitecore-content-hub-to-experience-commerce-connector.html	396384
Engine	Added a sample plugin that is delivered as part of the Engine SDK. The plugin defines the pipeline and blocks required for the Engine to register with the Content Hub's Azure Service Bus instance as a listener, to initiate synchronization queries to Content Hub, and to map the data to the sellable item properties. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/sample-sitecore-content-hub-to-experience-commerce-connector.html	377030 394021

Storefront Improvements

Features / Improvements – Storefront	ID
Added a new Storefront Bundle component rendering variant based on Scriban templating, providing a better developer experience by reducing the complexity of the rendering variant. This also provides support for optional bundled items and upgradable items. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-product-bundle-rendering.html	385667 386987
Added a new generic messaging service for improved developer experience. It allows better and uniform inter-component communication. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/rendering-communication.html	385870 358142
Added a new `RecommendedProducts` search token for looking up recommended products based on the last order placed. This can be used with the Commerce Search Results component to list recommended products. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/commerce-search-scopes.html	372529
Added a new embedded Scriban extension method `sc_xc_translate` that provides access to Commerce Terms under the Commerce Control Panel. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/the-embedded-commerce-functions-for-scriban-templates.html	364129
Added two new Scriban embedded objects `o_sitecontext` and `o_storefrontcontext` that provide access to `SiteContext` and the current catalog item. This is useful when the wildcard URL navigation approach is in use, and in connecting with canonical URLs. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/the-embedded-commerce-items-and-objects-for-scriban-templates.html	393191
Improved the Commerce Search Result component to support retrieval of price and stock information as part of the standard call to the SXA search service. This reduces the number of calls from 2 to 1, yielding better performance. However, this change may have an impact on performance where a CDN is in use. For more information see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/understanding-asynchronous-calls-and-caching.html	371651
Refactored the Product Price component into a new Scriban template based component, providing a better developer experience. The existing component has been retained for backward compatibility but marked deprecated. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-product-price-rendering.html	309840
Refactored the Minicart component into a new Scriban template based component, providing a better developer experience. The component now also uses the new messaging service to notify other components on the page that a cart line has been deleted and that changes have been made to the cart, so that content is refreshed.	377880

Features / Improvements – Storefront	ID
The existing component has been retained for backward compatibility but marked deprecated. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/configure-the-minicart-rendering.html	
Refactored the Shopping Cart Lines component into a new Scriban template based component, providing a better developer experience. The component now also uses the new messaging service to notify other components on the page that a cart line has been deleted and that changes have been made to the cart, so that content is refreshed. The existing component has been retained for backward compatibility but marked deprecated. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-shopping-cart-lines-rendering.html	278268
Refactored the Order Lines component into a new Scriban template based component, providing a better developer experience. The existing component has been retained for backward compatibility but marked deprecated. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/customizing-the-order-lines-rendering.html	406945
Integrated Babel transpilation into the Storefront solution, providing a better developer experience in writing JS and supporting different browser versions. This conforms to eslint rules for ECMAScript 6 and provides support for ES6 syntax in the Storefront code base. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/add-ecmascript-6--to-sxa-storefront-themes.html	318282
Commerce search related page events (`FacetApplied`, `SortorderApplied`) are now triggered when SXA Search components are used, including the Commerce Search Results component. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/extending-the-commerce-search-results-rendering-with-analytics.html	355653
A new Relevance option has been added to the Commerce sort options under the Sort Results component. It is now the default sort option when the customer visits the site. It is configured to use the Relevance sort direction, which means it honors boosting rules. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/create-and-configure-a-sort-order-for-a-list-of-products.html	373729

Commerce Connect Improvement

Features / Improvements – Commerce Connect	ID
Introduced a new processor for the New Order Placed pipeline. The processor flushes the current interaction with xDB and initiates the post-processing and population of calculated facets like `CommerceInteractionsCache` that store data about orders placed. This is useful when showing recommendations to the customer based on their last order placed within the same session.	400889

Features / Improvements – Commerce Connect	ID
The processor is disabled by default because of the possible impact; it splits the current session into multiple interactions in xDB. When not enabled, the recommendations based on last order will not display until the next visit/session because the interaction will not be processed and data will not be stored in the calculated facet until the session ends.	

Business User Experience Improvements

Features / Improvements – Business User Experience	ID
Improved the navigation within the Inventory Management interface to make it easier to edit inventory details. The user is now able to view inventory details directly by selecting a sellable item listed in the Inventory Management interface, whereas previously selecting the sellable item would navigate to a sellable item details page. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-inventory.html	347231
Improved the navigation when creating a new item in the Business Tools by redirecting to that new item upon creation, whereas previously when the item was created the user had to search for the created item. This enhancement is enabled by default on the Authoring environments of Habitat and AdventureWorks. It is disabled on the Shops environments because items created in the Shops environment are not published, so redirect does not apply.	372675
Added the ability to filter on the current catalog when associating sellable items to a category. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-a-sellable-item.html	345850 415126
Added a search restriction as part of bundle item selection, to restrict the search to items to within the current catalog. This avoids users from getting error messages when they try to associate sellable items from other catalogs.	388473
Added the ability to display in the Business Tools if a sellable item is contained in a bundle. Added a new `SellableItemToBundle` relationship.	391304
Added the ability for the merchandiser to target a promotion to the specific currency being used in the current context of the customer's session. This capability improves the usability in setting up promotions for multi-currency environments. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/promotions-qualifications.html	271011
Added validation upon deleting a sellable item from the catalog. If the sellable item is part of a bundle, the user is prompted to disassociate the sellable item from the bundle first. For more information, see: https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-a-sellable-item.html	406775
Implemented UI improvements to the PriceCard Snapshot `page` to more clearly describe how quantity and pricing is defined. For example, the Quantity defines the number of items in a cart when the price should be applied, and Price defines a new Sell Price for just one item. For more information, see:	342966

Features / Improvements – Business User Experience	ID
https://doc.sitecore.com/users/100/sitecore-experience-commerce/en/work-with-price-cards.html	

Performance and Scalability

Area	Features / Improvements – Performance and Scalability	ID
Generic Search	Added a new API endpoint in the Engine enabling provider-agnostic search, so that the user of this API does not need to know whether XC is configured to work with a specific search provider. In addition to providing a more generic capability, this improvement enables the Pricing, Promotions discovery performance improvements (see 378595 below). For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/generic-search-framework.html	379266 379916
Search	Improved the handling of throttled requests for Azure Search read operations. Implement restart if requests are throttled causing multiple exceptions.	407651
Search	Rectified how the Catalog and Cart controllers queried for data. They now correctly query the cache, instead of the previous behavior of querying the SOLR or Azure search services. This improvement reduces the likelihood that the search service becomes overloaded when the system is loaded with heavy traffic.	418772 418773
Indexing	Continued from XC9.3… further improved indexing performance by reducing the impact of catalog size on time to complete a full indexing operation.	385778
Incremental indexing	Improved the performance of incremental indexing, which helped Azure Search to function correctly when under heavy load.	418774
Mappings	Enabled Commerce to Content Tree mappings to be fetched in bulk from the Engine, instead of making individual calls which can result in many HTTP requests. This mapping occurs after indexing has completed, when updating the mappings on the CM and CD roles.	421177
Mappings	Prevented all catalogs being loaded into the mappings operation, when only one catalog was selected. Only the selected catalog needed to be loaded. This improvement could potentially significantly reduce the time to complete the mappings operation, and reduce the memory consumption on the CM role.	423535
Mappings	Updated the `CheckTemplatesMapping()` method to check if the template had been loaded within the lock. In a scenario where there were multiple requests coming to the lock after the lock was released, the first request entered the lock and called `InvokeHttpClientGet` before the template was loaded. This resulted in a call to load the templates that could have been avoided, and in some cases resulted in over 5% of threads being blocked.	426656
Bulk caching	Added support for bulk caching operations in the Sitecore caching Framework libraries. Added support for the MGET bulk get operation including updating `Sitecore.Framework.Caching.Redis` to a version that supports MGET. This enables the Pricing, Promotions discovery performance improvements (see 378595 below).	391534
Sliding cache	Improved the sliding cache behavior so that more frequently used items tend to remain within the cache, versus stale items. This has the effect of reducing the	412856

Area	Features / Improvements – Performance and Scalability	ID
	effect of clogging up the cache with items that seldom change, and minimizing spikes in response times when looking up more active items. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/enable-sliding-cache-expiration-in-commerce-engine-connect.html	
Redis	Reduced occurrences of `RedisTimeoutException` errors in the Storefront when under load, mostly from the home page submission of "/sxa/search/results" requests.	403455
PurgeCarts Minion	Improved the performance of the `SelectListEntitiesByRange` procedure when there is a large amount of data, which helped the `PurgeCartsMinion` to function correctly when under heavy load.	418943
Languages	Reduced the amount of SellabItem `GetCatalogItemAllLanguages` calls to Shops, and the subsequent queries to SQL. This improvement reduced the negative impact on response times and throughput for calls to Shops while mapping updates were being calculated, when a large catalog was configured and when under heavy load.	421525
Pricing, Promotions discovery	Formerly, runtime requests to the Engine for calculating pricing or promotions initiated SQL queries that returned all the pricing or promotion entities, and then calculated based on processing all the returned entities. Leveraging the new Generic Search (above), the discovery of price cards and snapshots and promotions has been modified to search the index, then calculate on a reduced subset of returned entities. Initial testing indicates that discovery plus calculation times are reduced by over 85% when there are around 6,000 price cards & snapshots in the database, Similar improvements were observed for promotions.	378595 372055
Promotions	Improved the performance in calculating promotions by merging the pipelines that calculate the line and cart level promotions. This improvement also improves the usability by making it easier to understand the application of promotions, for the most common use scenarios. Caution: breaking change – may alter existing promotions calculations. This improvement goes together with the new Promotions Prioritization feature (see 385560 earlier in this document).	373474
Promotions	Prevent promotions from being loading when the cart is empty. This can reduce the number of unnecessary calls to Redis when there as many promotions configured in the system.	388597
Cart API	Added the capability to add multiple cart lines at once. The new `AddCartLines` API calculates the cart only once, and not for each line item added. Added a corresponding Postman sample. This improvement was driven in part by the new Promotion - Free Gift with Purchase feature, by allowing multiple free gift lines to be added at the same time. Caution: breaking change - the signature has been changed to make use of `AddCartLinesCommand` instead of `AddCartLineCommand`. For more information, see: https://doc.sitecore.com/developers/100/sitecore-experience-commerce/en/cart-commands-and-pipelines.html	388586

Area	Features / Improvements – Performance and Scalability	ID
Customers	Reduced the time to load the Business Tools' Customers dashboard, when there is a large number of registered customers stored in the database, for example over 5 million customers.	413889

Framework and 3rd Party Software

Features / Improvements – Framework and 3rd Party Software	ID
Rebased on .NET 4.8	385781
Updated to .NetStandard	396388
Updated OData to the official release version of Microsoft.AspNetCore.OData	383989 399608
Updated .NET Core from 2.1 to 3.*	401801
Updated to SOLR 8.4 and SOLRNet 1.0.19	385815

Sample Data

Features / Improvements – Sample Data	ID
Added the ability to deploy the Commerce Engine without requiring the Habitat or AdventureWorks sample data.	270595
Added sample static and dynamic bundles to the Habitat Catalog sample data.	277770 373181
Added sample free gift with purchase promotions to the Habitat promotion book sample data.	399911

Resolved Issues

Area	Resolved Issues	ID
Catalog	Data import could fail if the Relationship Definition belonged to another Catalog. Example: create two Catalogs each using the same Relationship Definition entity; export both to a zip package; import each one by one. Result: import of the second Catalog failed with an error related to the Relationship Definition.	377022 377028
Catalog	The processing of `CatalogSystemIntervalAsynchronousStrategyBase` compared Local Time with UTC, which resulted in delays in processing.	401930 401932
Catalog	The processing of `CatalogUpdateMappings` incorrectly retained outdated versions of the Entities.	404473 404477
Catalog	Association and dissociation of a child entity to a parent only considered the version of the parent; it did not correctly consider the version of the child item.	406355 404477
Catalog	The rendering of Related Items could corrupt the relationship links for products associated to more than two catalogs or storefronts.	408670 408671
Catalog	Orphaned Categories and SellableItems could cause a NullReferenceException when they were accessed or indexed.	402728, 404293
Catalog	Child items such as Categories in the same level could disappear from the Storefront navigation, after modifying another unrelated item.	385595
Catalog	Product catalog item and catalog content was missing in web after publishing the Storefront.	278778
Habitat sample Catalog	The delivered sample Habitat catalog contained duplicate variants, which were causing error message, "An invalid variant combination has been selected" when browsing sellable items on a Storefront site.	270907 336758 376734
Merchandising	Updating data templates from the Content Editor created templates with new GUIDs. For example, in the Bus Tools create a Composer Template and assign it to all items in the Catalog by assigning it to the SellableItem entity; now in Sitecore delete and recreate the Sitecore templates using the buttons on the Commerce tab and assign it to a template; repeat this a second time. Result: the recreated template has a different GUID, so it cannot be resolved.	276416 315713
Merchandising	An insufficient error message was displayed when trying to add an existing category to a new catalog.	282044
Merchandising	The quantity was not localized in the Business Tools - Pricing table on the PriceSnapshotDetails page.	300105
Merchandising	Composer Templates with special characters in the view display name resulted in breaking the Update Data Templates, in some scenarios.	312087
Merchandising	The `SellableItemToBundle` list was incorrectly removed after disassociating one of several variants from the bundle. This list should have remained in the RelationshipLists, and only be removed after all its variants have been disassociated.	378207
Merchandising	It was not possible to delete an image from a variant if a sellable item did not have an image assigned.	379961
Merchandising	A Static Bundle containing both a physical and a digital sellable item could not be processed correctly.	381056 334357

Area	Resolved Issues	ID
Merchandising	The `BundleToSellableItem` relationship did not get removed from the database after the bundle was deleted.	404896
Merchandising	Location data was missing from a SellableItem in the Business Tools, pertaining to a language that was just added.	388463
Promotions	Users were unable to create a custom qualification when the input field was the DateTime Picker.	282057
Promotions	Applying several promotions or coupons with the Get Free Shipping benefit could result in the Total Discount being larger the Shipping cost.	355178
Promotions	A customer who had not previously made an order was not able to gain a benefit using the `CurrentCustomerOrdersTotalCondition`.	313678
Promotions	A private coupon allocation could be created even if the operation had been canceled.	384011
Promotions	Applying a promotion to a Cart with a higher value than cart total resulted incorrectly with a negative subtotal.	410423
Promotions	Coupons from a Promotion Book that had not been associated with a Catalog could be added to a Cart, whereas this should not have been possible.	415565
Promotions	The Edit End Date operation was enabled for a disabled promotion, whereas this operation should not have been permitted.	414260
Promotions	English property values of Promotions disappeared after moving the Promotion between workflow states, from Request Approval to Reject.	384987
Entitlements	The `KnownEntitlementsTags` policy was not taken into account because the entitlement tag was hardcoded in the `FilterCartLineFulfillmentOptionsBlock`.	277156
Fulfillment	Fulfillment fees were incorrectly calculated in multi-currency scenarios.	380621
Returns	The Refund transaction amount was incorrect in certain situations when performing multiple returns on same order, with federated payments.	390816
Indexing	The operation of the incremental and full Catalog Indexes resulted in different results, for unpublished Catalog versions.	350171
Indexing	The incremental and full index minions were incorrectly configured to run at the same time. For example, Pricing incremental and full index minions manipulate the same type of entities so if they ran at the same time, concurrency errors could potentially occur.	419210
Indexing	The full path to the child entity in the index field handler was not calculated correctly if multiple versions of a parent entity existed.	409570 409753
Indexing	Switch-on-release for Solr in Experience Platform indexing was not functioning correctly.	411076
Indexing	Commerce index implementations were not fully aligned between Solr and Azure. For example, the `ArtifactStoreId` and `DateCreated` fields in Azure were incorrectly marked as `IsRetrievable = false` so these fields were missed in Azure.	381489
Indexing	Errors may occur in the Commerce Engine command `AzureContextCommand.QueryDocuments.Index:PriceCardsScope` when the system is heavy loaded with traffic.	403450
Indexing	The threshold for cleaning all caches on the `indexing:completed` event handler was not configurable.	410145 304030

Area	Resolved Issues	ID
	It had a hardcoded check if number of changes is greater than 20. If so all Sitecore caches are cleared, this created a performance degradation for Sitecore instances until the caches were populated again. A setting of 20 could be small in some instances, and it could not easily be changed through configuration.	
Caching	The `GetItemByPathPipeline` could create inconsistent cache records.	385005
Caching	Unpublished Sellable Items data could disappear from Content Management, and the Business Tools, due to NegativeCache not functioning correctly.	386054, 386263
Caching	The `CommercePublishCacheRefresh` handler of the `publish:end:remote` event failed on the CD instance, when a targeted database had different names between the CD and CM.	342364
Caching	A `CacheRefreshEvent` event was incorrectly still being raised from the `RemoveItemFromSitecoreCaches` method.	388481
Mappings to Platform	Transient errors accessing Mapping info from the Shared Database could cause short term downtime when an Azure App Service deployment was scaled out.	396731 396734
Redis	The Commerce Engine did not gracefully handle Redis failure scenarios, for example the Engine failed to persist entities. Improvements enabled the Engine to continue to function correctly even when Redis failed.	395155
Minions	Minions returned the status code of "OK" when a failure occurred, and should have reported that error(s) occurred with a proper status code.	282048
Content item	A Storefront item was not always retrieved from the ContentEntities table even though it existed. Instead an InvalidShop error appeared, "Shop 'CommerceEngineDefaultStorefront' does not exist."	409282 358019
Import	Import of relationships failed due to the path separator. Different zip archiving tools may produce different paths inside the archive. The path to the relationships folder in an import file was assigned an unexpected path separator (/ vs \), which caused it to be picked up by the relationship import and throw exceptions. Note that this only affected the relationships, as all other files were stored in the root of the archive.	425202
Storefront	When placing an order from an abandoned cart in a newly created site, the product details were missing from the abandoned cart email.	382582
Storefront	For anonymous customers, an email was not sent to the user when an order was placed, or when the shopping cart was abandoned. The cause was the contact was not enrolled in the Marketing Automation plans due to an issue with xDB Tracker.	381952 384728
Storefront	`BulkManager.GetSellableItemSummary` added an extra slash to the URL request to the Commerce Engine, which broke pages containing the Commerce Search Results component.	395221
Storefront	The pipeline processor `TriggerOrderOutcome` had been incorrectly commented out, due to an error in the Experience Platform code. This processor is now correctly executed from the pipeline `commerce.orders.submitVisitorOrder`.	349119
Storefront	Two commerce scripts `cart-minicart-model.js` and `catalog-productvariant-model.js` used the self variable without the `var` keyword, effectively breaking custom js since it becomes global.	387747
Storefront	In the add to cart component, the quantity could not be changed after the user had entered a number. It was not required to have both buttons provided by the	276561

Area	Resolved Issues	ID
	brower and buttoms provided by the component, so the browser provided buttons have been hidden using CSS.	
Storefront	Method `BulkManager.GetBundleSelectionPrice` was incorrectly tagged with the attribute `Obsolete`. This method is used by the Product Bundle component and should not have been obsolete.	406967
Storefront	Some JS scripts in commerce components theme used the "self" without `var` keyword, which broke some of custom js since it becomes global. The `var` keyword is now used instead of self.	387747
Storefront	The Storefront branded theme caused an issue in the Forms checkbox validation.	389363
Storefront	The caching option `VaryByUrl` when enabled incorrectly overrode other caching options. It now functions together with other caching options.	409667
Storefront	The canonical URL generated by the Commerce Metadata component used only the HTTP protocol and ignored the protocol in use for the requested page.	397335
Storefront	The shopper was unable to select the + icon in quantity box to increase the quantity of products to add to cart.	276981
Storefront	An `ArgumentOutOfRangeException` error message displayed when adding a query parameter with no value in the URL.	421385 421384
Commerce Connect	An `ArgumentNullException` error message displayed in certain situations while creating a new Commerce customer.	417328 421010
Commerce Connect	The `ConvertPageEventDataToCartPageEvent.CreateEvent` operation would return a null instead of the real event.	234527
Deployment	Was unable to create new categories or sellable items after running the `CleanEnvironment` Postman action.	323847
Deployment	Search processes ran during environment initialization, whereas they should not have.	415713
Deployment	The Minion policy had the wrong namespace.	415967
Deployment	The role `required` for `RemoteEventMap` and `ClientDataStore` was outdated.	401962
Deployment	The content item was not retrieved from the `ContentEntities` table, even though it existed.	413144
Deployment	The Storefront item was not retrieved from the `ContentEntities` table, even though it existed.	358019
Deployment	After upgrading Commerce Connect, Storefront items in the Control Panel got overwritten, which resulted in losing some Storefront configuration.	210978
Deployment	Issues experienced in deploying Commerce when redis is deployed on a separate host.	388602
Deployment	Redundant files were present on ma-ops and ma-rep roles in an Azure deployment.	380246
Deployment	BootStrapping Commerce Services failed intermittently in a Cloud deployment. Added retry logic to Engine bootstrapping.	426616
Deployment	SQL timeout issues occurred during deployment. Timeouts were intermittent. Improved logging, changed the protocol for connection to Sitecore to HTTPS, added wait time between redis connection attempts, ensured that long running commands completed in time.	402021
Upgrade scripts	Former release upgrade scripts were included in the release package, which in at least one scenario caused confusion in what upgrade paths were supported in the	411502

Area	Resolved Issues	ID
	release. The resolution was simply to remove non-applicable former upgrade scripts.	
Postman sample	The Postman sample "Associate Sellable Item to Inventory Set" incorrectly use the `AssociateSellableItem` action, whereas it was supposed to use the `AssociateSellableItemToInventorySet` action.	374201

Known Issues

Area	Known Issues	ID
SXA Performance	We recommend you install SXA 10.0.0 hotfix to the "Scriban errors in logs: known issue. The hotfix can be found here: https://kb.sitecore.net/articles/285179 This hotfix resolves a performance degradation that can occur when the system is under heavy load; a race condition can occur in Scriban-based rendering variants.	427155
Minions	Minions are not successfully rebuilding search indexes during deployment - causing price calculation issues. This issue is observed intermittently, and cannot yet be reliably reproduced. Work around: manually rebuild the indexes.	421490
Storefront	Missing Scriban templates are missing when exporting the site or page using Creative Exchange. The Scriban template is missing from the exported zip package.	422872
Storefront	Unable to select a comparison for an inventory rule. In the Experience Editor, select a component to personalize. In the Create Rule dialog box, select the Current product has stock status equals to specific stock status in location equals to specific location. The user is not able to select the comparison (equals to), and upon selecting a specific stock status an error message is displayed saying that you must select the comparison first.	378153
Storefront	Deleting a Commerce Catalog folder item or removing a site and restoring it from the Recycle Bin, will incorrectly have catalogs selected but not showing. For example, using the Habitat sample Catalog, delete an item in the Content Editor… '/sitecore/content/Sitecore/Storefront/Home/Catalogs' while the Habitat catalog is still selected/associated. Launch the Recycle Bin and restore the Catalogs item, and select the catalogs item. The result is that the selected and associated catalogs are not displayed beneath the catalogs item Habitat_Master.	380355
Storefront	Unable to select the Habitat_Master catalog in Catalog Configuration present under Commerce Control panel for a new site. The result is that even though Habitat_Master is selected, the catalog field is not populated. To work around the issue, uncheck the catalog associated with site -> Save and associate the catalog again, and Save.	380566
Storefront	The second language selectors are displayed on the Storefront site after resizing the browser window to a smaller size, whereas the selectors are supposed to be removed in a smaller browser size.	381118
Storefront	Some labels and text on Storefront are not localized. This includes: the Quantity text in the Minicart, Edit account details, Order status, Print Order button, the tooltip… Register new account and checkout as guest, Info/Error in the message header	381158
Storefront	When sending out product recommendations via mail, an infinite loop scenario could occur when more than one product is purchased. If less than the desired and specified number of recommended products is available in the specified relationship field for the purchased products, it may loop indefinitely, while trying to fetch the specified number of recommendations.	400891

Area	Known Issues	ID
Storefront	Customers with an account that places orders as anonymous will not get product recommendations from the latest order by mail through the New Order Placed marketing automated campaign.	400893
Storefront	On bundles with only optional items, the shopper is not able to proceed to add an item to the cart if no item is selected. The Add-to-Cart button should be disabled until at least one item has been selected.	407937
Storefront	The merchandiser is incorrectly able to configure a sellable item and one or more variants of the same sellable item in the same free gift collection. This could lead to undefined behavior in the back end calculations. The merchandiser should instead be prevented from configuring multiple free gifts based on the same product with variants.	409322
Marketing Automation	Unable to activate a campaign in a non-English tier-1 language.	377996

Breaking Changes

This section includes sub-sections:

Sitecore Experience Commerce 10.0 upgrades the .NET Framework to .NET Standard, and introduces upgrades to .NET Core and other lower layer software. See the **Features/Improvements - Framework and 3rd Party Software** section above for more information. These upgrades introduce a large number of breaking changes. See also the Breaking Change Reports contained in a zip file that can be downloaded from the Commerce 10 download page:
https://dev.sitecore.net/Downloads/Sitecore_Commerce/100/Sitecore_Experience_Commerce_100.aspx
These breaking change reports for each Engine plugin compare a recent XC10.0 build to XC9.3.

API or Behavior Changes in Experience Commerce 10.0

Area	API or Behavior Changes	ID
Engine SDK	Visual Studio 2017 is no longer supported as a result of upgrade to .NET Core 3.1. Vsix does not work with .NET Core 3.1; it is replaced with the 'dotnet new' command used to install a custom project template.	401508
Engine SDK	Updated target framework to 3.1. Extensions of the Commerce SDK now need target the new runtime.	401801
Controllers	All Controllers are now `CommerceODataController`	N/A
ItemModel	`Sitecore.Services.Core.Model.ItemModel` type has been replaced with `Sitecore.Commerce.Plugin.Management.ItemModel`	N/A
Add Cart	`DoActionAddLineItemBlock` - The signature has been changed to make use of `AddCartLinesCommand` (plural) instead of `AddCartLineCommand`.	373474
Merged cart calculations	Merge `ICalculateCartLinesPipeline` and `ICalculateCartPipeline` - The total is now recalculated after each action, so some existing Promotions that were applied before will no longer be applied. And, there will now only be one Exclusive Promotion, whereas previously there could be one Exclusive Promotion for Cart Lines and another for the Cart.	373474
Delete Cart	`DeleteCart`(string cartId) has been added as a new API	N/A
Versioning	Removed versions for catalog hierarchy and relationships. Most entity relationships are not versioned in 10.0. Only these 2 relationships still respect entity version: `BUNDLETOSELLABLEITEM` and `BUNDLETOSELLABLEITEMVARIANT`. So a bundle in different versions can have different related items. Any other relationship will be the same for each version. For example, let category v1 has 5 sellable items. Category v1 is published and items are visible on the storefront site. Create the next version of the same category v2 and remove a related sellable item. Even the category v2 is	412844

Area	API or Behavior Changes	ID
	not published yet, the item will be removed for the site as category v1 and v2 have the same relationships.	
Promotions	Promotions may be returned for the sellable item by default for the following APIs: `GetSellableItemsSummary`, `GetBulkPrices`, `SellableItems.Get` To disable this, pass the `IgnorePromotions` header.	N/A
Storefront languages	Tier-1 languages are no longer included as optional modules that can be selected during site scaffolding. The content for each of the languages is no longer part of the branch templates for the site template. This is in line with SXA best practice. Adding a language to a site is now supported through the standard SXA functionality for adding new languages. Storefront Language files can be downloaded from the Commerce 10 download page.	402264
Azure Search	`Sitecore.Commerce.Plugin.Search.Azure.AzureContext.GetSearchIndexForQueriesClient` => is not longer static.	N/A
Action(post)	`Bootstrap()` is now an action(post)	N/A
Action(post)	`EnsureSyncDefaultContentPaths()` is now an action(post)	N/A
Deployment	A new mandatory parameter (`'sampledata'` with true/false value) is added for `InitializeEnvironment`() API. It is used to specify whether sample data (e.g. Habitat/AWC catalogs, promotions, price cards etc.) will be created during `InitializeEnvironment`. The deployment scripts and Postman sample have been updated to include the new parameter and set to "true" as default (existing behavior).	270595

Obsoleted/Deprecated in Experience Commerce 10.0

Code obsoleted or deprecated in this release will generally be removed in the next Experience Commerce 10.1 release.

Obsoleted/Deprecated in Experience Commerce 10.0
PaaS Deployments: Azure App Service (PaaS) deployments will be discontinued as of the next Commerce 10.1 release. It is intended that Containers deployments will fully replace PaaS deployments starting in Experience Commerce 10.1.
Sitecore.ContentSearch.Azure: Azure Search is deprecated in Commerce 10.0 and will be removed in Commerce 10.1. If you are starting a new Sitecore project, we recommend that you use Solr as your search engine.
Sitecore Commerce Core: - Sitecore.Commerce.Core.CommerceController => Use CommerceODataController instead. - Sitecore.Commerce.Core.CoreExtensions.RemoveOdataTypeProperties => No longer required as the new OData version is more type-safe. - Sitecore.Commerce.Core.ODataFormInputFormatter => No longer required with AspNet Core OData. - Sitecore.Commerce.Core.PostFindEntityPipeline => Not longer in use - Sitecore.Commerce.Core.PostFindEntitiesBlock => Not longer in use

Obsoleted/Deprecated in Experience Commerce 10.0
- Sitecore.Commerce.Core.RefreshEntitiesInCacheBlock.CacheEntities(PersistEntityArgument argument, CommercePipelineExecutionContext context) => Use CacheEntities(List<CommerceEntity> entities, CommercePipelineExecutionContext context) instead. - Sitecore.Commerce.Core.RefreshEntitiesInCacheBlock.CacheEntity(CommerceEntity entity, EntityCachingPolicy cachingPolicy, CommercePipelineExecutionContext context) => Use CacheEntities(List<CommerceEntity> entities, CommercePipelineExecutionContext context) instead.
Management plugin: - Sitecore.Commerce.Plugin.Management.ItemModelExtensions => This extension class is no longer used, use the Sitecore.Commerce.Plugin.Management.ItemModel class directly.
Views plugin: - Sitecore.Commerce.Plugin.Views.CommandControllers.DoUxAction => Use DoAction instead. - Sitecore.Commerce.Plugin.Views.DoActionPaginateCollectionBlock.Commander => Use AsyncDoActionPaginateCollectionBlock instead. - Sitecore.Commerce.Plugin.Views.DoActionPaginateListBlock.GetEntities => Use AsyncDoActionPaginateCollectionBlock instead. - Sitecore.Commerce.Plugin.Views.ViewsConstants.PopulateEntityActions => No longer in use.
Catalog plugin: - Sitecore.Commerce.Plugin.Catalog.CatalogsController.GetCatalogConnect => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.ApiController.GetCatalogConnect => Use GetCatalogItemAllLanguages(type, id) instead. - Sitecore.Commerce.Plugin.Catalog.CategoriesController.GetCategoryConnect => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.ApiController.GetCategoryConnect => Use GetCatalogItemAllLanguages(type, id) instead. - Sitecore.Commerce.Plugin.Catalog.SellableItemsController.GetSellableItemConnect => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.ApiController.GetSellableItemConnect => Use GetCatalogItemAllLanguages(type, id) instead. - Sitecore.Commerce.Plugin.Catalog.SellableItemsController.GetSellableItemsConnect => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.ApiController.GetSellableItemsConnect => Use GetCatalogItemsAllLanguages(type, listName, skip, take) instead. - Sitecore.Commerce.Plugin.Catalog.SellableItemsController.GetSellableItemsByParent => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.SellableItemsController.GetSellableItem => moved to proper controller Sitecore.Commerce.Plugin.Catalog.ApiController - Sitecore.Commerce.Plugin.Catalog.BaseCalculateSellPriceBlock.ResolvePriceCardsByBook - Use SearchSnapshotIdsByTags and GetSnapshotsFromSearchResults instead - Sitecore.Commerce.Plugin.Catalog.BaseCalculateSellPriceBlock.FilterPriceSnapshotsByTags - Use SearchSnapshotIdsByTags and GetSnapshotsFromSearchResults instead - Sitecore.Commerce.Plugin.Catalog.DeleteSitecoreItemTombstonesCommand => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.ApiControllers.DeleteSitecoreItemTombstones => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.SitecoreItemTombstone => No longer required. Use Sitecore.Commerce.Core.Tombstone instead. - Sitecore.Commerce.Plugin.Catalog.DeleteSitecoreItemTombstonesArgument => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.DeleteSitecoreItemTombstonesBlock => No longer required since SitecoreItemTombstone is no longer in use.

Obsoleted/Deprecated in Experience Commerce 10.0
-Sitecore.Commerce.Plugin.Catalog.BaseIndexDeletedSitecoreItemBlock.CreateTombstone (CommercePipelineExecutionContext context, CommerceCommander, DeleteEntityArgument, List<SitecoreCatalogIndexingPolicy> policies) => Use AddEntityToSitecoreIndexDeletedList(CommercePipelineExecutionContext context, CommerceCommander, DeleteEntityArgument, List<SitecoreCatalogIndexingPolicy> policies) instead. - Sitecore.Commerce.Plugin.Catalog.IndexDeletedRelationshipBlock => No longer required, since we are not using Tombstones when indexing - Sitecore.Commerce.Plugin.Catalog.IDeleteSitecoreItemTombstonesPipeline => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.IIndexDeletedEntitiesPipeline => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.IIndexDeletedEntityPipeline => No longer required since SitecoreItemTombstone is no longer in use. - Sitecore.Commerce.Plugin.Catalog.IndexDeletedEntitiesPipeline => No longer in use. - Sitecore.Commerce.Plugin.Catalog.IndexDeletedEntityPipeline => No longer in use. - Sitecore.Commerce.Plugin.Catalog.ValidateSellableItemBlock => Use the plural version ValidateSellableItemsBlock. - Sitecore.Commerce.Plugin.Catalog.CatalogConstants.GetRelationshipDefinitionPipeline. - Sitecore.Commerce.Plugin.Catalog.CatalogConstants.ValidateSellableItemBlock. - Sitecore.Commerce.Plugin.Catalog.DoActionEditBundleBlock => was renamed to DoActionEditStaticBundleBlock. - Sitecore.Commerce.Plugin.Catalog.DoActionSellableItemDisassociateBlock => The constructor DoActionSellableItemDisassociateBlock(CommerceCommander commerceCommander,IPersistEntityPipeline persistEntityPipeline) is obsolete. Use DoActionSellableItemDisassociateBlock(CommerceCommander commerceCommander) instead. - Sitecore.Commerce.Plugin.Catalog.SellableItemPricing => The constructor SellableItemPricing(SellableItem sellableItem) is obsolete. Use SellableItemPricing(SellableItem, string variantId) instead. - Sitecore.Commerce.Plugin.Catalog.BaseCalculateSellPriceBlock.ResolveSnapshotByTags(IList<Tag>, CommercePipelineExecutionContext) => Use ResolveSnapshotByTags(SellableItem, CommercePipelineExecutionContext, ItemVariationComponent) instead. - Sitecore.Commerce.Plugin.Catalog.BaseCalculateSellPriceBlock.ResolvePriceCardsByBook (CommercePipelineExecutionContext) => Use ResolveSnapshotByTags(SellableItem, CommercePipelineExecutionContext, ItemVariationComponent) instead. - Sitecore.Commerce.Plugin.Catalog.BaseCalculateSellPriceBlock.FilterPriceSnapshotsByTags(IList<PriceCard>, IList<Tag>, CommercePipelineExecutionContext) => Use ResolveSnapshotByTags(SellableItem, CommercePipelineExecutionContext, ItemVariationComponent) instead. - Sitecore.Commerce.Plugin.Catalog.RelationshipExtensions.GetRelationshipListName(this string relationshipName, string sourceEntityFriendlyId, int sourceEntityVersion) = > use GetRelationshipListName(this string relationshipName, string sourceEntityFriendlyId) instead. - Sitecore.Commerce.Plugin.Catalog.CloneCatalogArgument.CatalogEntityVersion => no longer used. - Sitecore.Commerce.Plugin.Catalog.CreateRelationshipBlock => Use CommerceCommander to resolve pipelines. - Sitecore.Commerce.Plugin.Catalog.DeleteRelationshipBlock => Use CommerceCommander to resolve pipelines. - Sitecore.Commerce.Plugin.Catalog.UpdateCatalogHierarchyBlock => Use CommerceCommander to resolve pipelines. - Sitecore.Commerce.Plugin.Catalog.DoActionDisassociateBlock => Constructor was changed. Use constructor DoActionDisassociateBlock(IFindEntityPipeline findEntityPipeline, AssociateItemToParentBlock, IDisassociateSellableItemFromParentPipeline disassociateSellableItemFromParentPipeline, FindEntitiesInListCommand findEntitiesInListCommand). - Sitecore.Commerce.Plugin.Catalog.DoActionSellableItemDisassociateBlock => Constructor was changed. Use constructor DoActionSellableItemDisassociateBlock(CommerceCommander commerceCommander). - Sitecore.Commerce.Plugin.Catalog.GetPromotionItemDetailsViewBlock => Constructor has changed. Use constructor GetPromotionItemDetailsViewBlock(IGetSellableItemPipeline getSellableItemPipeline).

Obsoleted/Deprecated in Experience Commerce 10.0
- Sitecore.Commerce.Plugin.Catalog.GetSellableItemDetailsViewBlock.AssociateSellableItemToParent => Method signature change. Use public void AssociateSellableItemToParent(CommerceEntity entity, CommercePipelineExecutionContext context, EntityView entityView). - Sitecore.Commerce.Plugin.Catalog.CloneRelationshipsBlock.CloneRelationships => Method signature change. Use CloneRelationships(CloneCatalogArgument arg, CommercePipelineExecutionContext context, string originalSourceFriendlyId, string clonedSourceId, RelationshipDefinition relationshipDef, bool ignoreNonClonedTargets = false). - Sitecore.Commerce.Plugin.Catalog.StreamBulkCatalogItemsToArchiveBaseBlock.GetRelationshipListName => Method has been removed.
Entity Versions plugin: - Sitecore.Commerce.Plugin.EntityVersions.VersionListReferencesBlock is obsolete => Use Sitecore.Commerce.Plugin.Catalog.VersionBundleListReferencesBlock instead.
Pricing plugin: - Sitecore.Commerce.Plugin.Pricing.PricingConstants.GetPricing - Sitecore.Commerce.Plugin.Pricing.PricingConstants.FilterPriceBookActionsBlock - Sitecore.Commerce.Plugin.Pricing.PricingConstants.FilterPriceCardActionsBlock - Sitecore.Commerce.Plugin.Pricing.PricingConstants.DoActionRemovePriceTagBlock
Promotions plugin: - Sitecore.Commerce.Plugin.Promotions.ApplyPromotionsBenefitsCommand => The constructor ApplyPromotionsBenefitsCommand(IApplyPromotionsBenefitsPipeline applyPromotionsBenefitsPipeline,IFindEntitiesPipeline findEntitiesPipeline,IFindEntityPipeline findEntityPipeline,IServiceProvider serviceProvider) should be used. - Sitecore.Commerce.Plugin.Promotions.ApplyPromotionsArgument => The constructor is obsolete, use ApplyPromotionsArgument(IEnumerable<Promotion> promotions) instead. - Sitecore.Commerce.Plugin.Promotions.ApplyPromotionsArgument => The PromotionsIds property is obsolete, use Promotions collection instead. - Sitecore.Commerce.Plugin.Promotions.DiscoverPromotionsArgument => The BenefitTypeFilter property is not in use after merging ICalculateCartLinesPipeline and ICalculateCartPipeline in one. - Sitecore.Commerce.Plugin.Promotions.EvaluatePromotionsArgument => The BenefitTypeFilter property is not in use after merging ICalculateCartLinesPipeline and ICalculateCartPipeline in one. - Sitecore.Commerce.Plugin.Promotions.CalculateCartLinesPromotionsBlock => Merged into the CalculateCartPromotionsBlock. - Sitecore.Commerce.Plugin.Promotions.SearchForPromotionsBlock(CommerceCommander) ctor - Use SearchForPromotionsBlock(SearchEntitiesCommand) constructor instead. - Sitecore.Commerce.Plugin.Promotions.FilterNotApprovedPromotionsBlock - Sitecore.Commerce.Plugin.Promotions.FilterPromotionsByBenefitTypeBlock - Sitecore.Commerce.Plugin.Promotions.FilterPromotionsByBookAssociatedCatalogsBlock - Sitecore.Commerce.Plugin.Promotions.FilterPromotionsByValidDateBlock - Sitecore.Commerce.Plugin.Promotions.GetPromotionsToApplyBlock => Constructor removed. Use default constructor.
Coupons plugin: - Sitecore.Commerce.Plugin.Coupons.DoActionPaginateCouponAllocationsBlock => public constructor no longer needed. - Sitecore.Commerce.Plugin.Coupons.AddCouponToCartBlock => Constructor signature change. Use AddCouponToCartBlock(IFindEntityPipeline findEntityPipeline, IGetBookAssociatedCatalogsPipeline getBookCatalogsPipeline).
Carts plugin: - Sitecore.Commerce.Plugin.Carts.IAddCartLinePipeline => Now use the plural version IAddCartLinesPipeline. - Sitecore.Commerce.Plugin.Carts.AddCartLinePipeline => Now use the plural version AddCartLinesPipeline. - Sitecore.Commerce.Plugin.Carts.AddCartLineBlock => Now uses the plural version AddCartLinesBlock. - Sitecore.Commerce.Plugin.Carts.CalculateCartLinesSubTotalsBlock => Merged into the CalculateCartSubTotalsBlock. - Sitecore.Commerce.Plugin.Carts.CalculateCartLinesTotalsBlock => Merged into the CalculateCartTotalsBlock. - Sitecore.Commerce.Plugin.Carts.ClearCartLinesBlock => Merged into the ClearCartBlock.

Obsoleted/Deprecated in Experience Commerce 10.0
- Sitecore.Commerce.Plugin.Carts.CalculateCartLinesPipeline => Merged into the ICalculateCartPipeline. - Sitecore.Commerce.Plugin.Carts.ICalculateCartLinesPipeline => Merged into the ICalculateCartPipeline. - Sitecore.Commerce.Plugin.Carts.PurgeCartsBlock => Not in Use. A new block was added in the SQL plugin. - Sitecore.Commerce.Plugin.Carts.PurgeCartsFromListBlock => Not in Use. A new block was added in the SQL plugin.
Orders plugin: - Sitecore.Commerce.Plugin.Orders.DelayedAvailability.ProcessWaitingForAvailabilityLinesBlock => Use IPopulateAvailabilityPipeline instead.
Inventory plugin: - Sitecore.Commerce.Plugin.Inventory.PopulateItemAvailabilityBlock => Use IPopulateAvailabilityPipeline instead. - Sitecore.Commerce.Plugin.Inventory.PopulateItemAvailabilityComponentBlock => Use IPopulateAvailabilityPipeline instead. - Sitecore.Commerce.Plugin.Inventory.DoActionEditInventoryInformationBlock => Constructor signature change. Use DoActionEditInventoryInformationBlock(EditInventoryInformationCommand editInventoryInformationCommand, AssociateSellableItemToInventorySetCommand associateSellableItemCommand, IFindEntityPipeline findEntityPipeline).
Availability plugin: - Sitecore.Commerce.Plugin.Availability.IPopulateItemAvailabilityComponentPipeline => Use IPopulateAvailabilityPipeline instead. - Sitecore.Commerce.Plugin.Availability.PopulateItemAvailabilityComponentPipeline => Use IPopulateAvailabilityPipeline instead.
Fulfillment plugin: - Sitecore.Commerce.Plugin.Fulfillment.CartFulfillmentsController => Methods were moved to the ApiController, this specific controller is no longer needed. - Sitecore.Commerce.Plugin.Fulfillment.FulfillmentMethodsController => Methods were moved to the ApiController, this specific controller is no longer needed. - Sitecore.Commerce.Plugin.Fulfillment.FulfillmentOptionsController => Methods were moved to the ApiController, this specific controller is no longer needed.
Payments plugin: - Sitecore.Commerce.Plugin.Payments.PaymentMethodsController => Methods were moved to the ApiController, this specific controller is no longer needed. - Sitecore.Commerce.Plugin.Payments.PaymentOptionsController => Methods were moved to the ApiController, this specific controller is no longer needed.
Search plugin: - Sitecore.Commerce.Plugin.Search.SearchNode - Use IGenericSearchNode<ISearchQueryVisitor, SearchQueryArgument> instead. - Sitecore.Commerce.Plugin.Search.SearchQuery(SearchNode) - Use ctor(IGenericSearchNode<ISearchQueryVisitor, SearchQueryArgument>) instead. - Sitecore.Commerce.Plugin.Search.SearchQuery.Node - Use SearchQuery.SearchNode instead. - Sitecore.Commerce.Plugin.Search.SearchQuery.GetQuery(ISearchQueryVisitor, SearchQueryArgument) - Use GetQuery(ISearchNodeVisitor, SearchQueryArgument) instead. - Sitecore.Commerce.Plugin.Search.ContainsOperator - Use ContainsNode instead. - Sitecore.Commerce.Plugin.Search.FieldName - Use FieldNameNode instead. - Sitecore.Commerce.Plugin.Search.FieldValue - Use FieldValueNode instead - Sitecore.Commerce.Plugin.Search.NotEqualOperator - Use NotEqualsNode instead. - Sitecore.Commerce.Plugin.Search.ISearchQueryVisitor - Use ISearchNodeVisitor instead. - Sitecore.Commerce.Plugin.Search.ISearchQueryVisitorFactory - Use ISearchNodeVisitorFactory instead. - Sitecore.Commerce.Plugin.Search.SearchQueryVisitorFactory - Use SearchNodeVisitorFactory instead. - Sitecore.Commerce.Plugin.Search.Azure.AzureSearchQueryVisitor - Use AzureSearchNodeVisitor instead. - Sitecore.Commerce.Plugin.Search.Solr.SolrSearchQueryVisitor - Use SolrSearchNodeVisitor instead. - Sitecore.Commerce.Plugin.Search.SearchViewPolicy.PageSize => This property is now obsolete. The global PaginationPolicy is always used throughout the system.

Obsoleted/Deprecated in Experience Commerce 10.0
- Sitecore.Commerce.Plugin.Search.SearchViewPolicy.PageSizeOptions => This property is now obsolete. The global PaginationPolicy is always used throughout the system.
Commerce Engine Connect: - Sitecore.Commerce.Engine.Connect.CatalogRepository.GetCatalogItemEntities(…) - Use GetCatalogItemEntitiesAllLanguages(…) instead. - Sitecore.Commerce.Engine.Connect.CatalogRepository.GetListItems(…) - Use GetListItems<TEntity>(string, string, int) instead. - Sitecore.Commerce.Engine.Connect.Events.CacheRefreshEvent => Not longer in use. - Sitecore.Commerce.Engine.Connect.DataProvider.Definitions.KnownItemIds.NavigationItemTemplateId => Template is not in use. - Sitecore.Commerce.Engine.Connect.KnownTemplateIds.CommerceNavigationItemTemplate => Template is not in use. - Sitecore.Commerce.Engine.Connect.Search.IsItemNavigation(Item item) => Navigation Item template is not in use. - Sitecore.Commerce.Engine.Connect.DataProvider.CanProcessItem(ItemDefinition itemDefinition, bool includeNavigationItemTemplate = true) => Use CanProcessItem(ItemDefinition itemDefinition) instead.

Removed in Experience Commerce 10.0

Removed code that was obsoleted in Experience Commerce 9.3. The list below is an excerpt from the Commerce 9.3 Release Notes. Refer to 9.3 Release Notes for more information.

Removed as of Experience Commerce 10.0
Commerce Core: - Sitecore.Commerce.Core.MinionBossPolicy(string, string, TimeSpan) - Sitecore.Commerce.Core.MinionPolicy(string, string, TimeSpan) - Sitecore.Commerce.Core.MinionPolicy(string, string, List, TimeSpan, int) - Sitecore.Commerce.Core.MinionPolicy.ListToWatch
Catalog plugin: - Sitecore.Commerce.Plugin.Catalog.CatalogItems - Sitecore.Commerce.Plugin.Catalog.GetMappingsForIdFromDbCommand.Process(CommerceContext commerceContext, string environmentName, string sitecoreId) - Sitecore.Commerce.Plugin.Catalog.BaseFormatComposerViewPropertyBlock.FormatSellableItem (SellableItem item, CommercePipelineExecutionContext context) - Sitecore.Commerce.Plugin.Catalog.GetSellableItemsCommand - Sitecore.Commerce.Plugin.Catalog.GetSellableItemsArgument()
Inventory plugin: - Sitecore.Commerce.Plugin.Inventory.GetInventoryCatalogPipeline - Sitecore.Commerce.Plugin.Inventory.GetInventorySkuPipeline - Sitecore.Commerce.Plugin.Inventory.GetInventoryItem - Sitecore.Commerce.Plugin.Inventory.GetInventoryInformationPipeline - Sitecore.Commerce.Plugin.Inventory.PopulateInventoryItemBlock - Sitecore.Commerce.Plugin.Inventory.LoadItemInventoryItemsBlock - Sitecore.Commerce.Plugin.Inventory.PopulateLineItemsInventoryBlock - Sitecore.Commerce.Plugin.Inventory.LoadInventoryItemBlock - Sitecore.Commerce.Plugin.Inventory.GetCartLinesInventory - Sitecore.Commerce.Plugin.Inventory.PopulateInventoryItemAllocationBlock - Sitecore.Commerce.Plugin.Inventory.GetEditInventoryInformationViewBlock
Other plugins:

Removed as of Experience Commerce 10.0
- Sitecore.Commerce.Plugin.Coupons.PersistCouponBlock - Sitecore.Commerce.Plugin.Coupons.PersistPrivateCouponGroupBlock - Sitecore.Commerce.Plugin.Orders.PersistOrderBlock - Sitecore.Commerce.Plugin.Carts.PersistCartBlock - Sitecore.Commerce.Plugin.Search.Solr.SolrContextCommand.DeleteDocuments(string name, CommerceContext context) - Sitecore.Commerce.Plugin.Search.SearchScopePolicy(string, string, bool, bool, bool, string, string, string, string, string, List) - Sitecore.Commerce.Plugin.Search.SearchScopePolicy.FullListName Sitecore.Commerce.EntityViews.GetListViewBlock
Commerce Engine Connect Search: - Sitecore.Commerce.Engine.Connect.Search.ComputedFields.CatalogEntityIdField - Sitecore.Commerce.Engine.Connect.Search.ComputedFields.CommerceAncestorIdsField - Sitecore.Commerce.Engine.Connect.Search.ComputedFields.CommerceHasChildrenField - Sitecore.Commerce.Engine.Connect.Search.ComputedFields.CommerceSearchItemTypeField - Sitecore.Commerce.Engine.Connect.Search.ComputedFields.ExcludeFromWebsiteSearchResultsField - Sitecore.Commerce.Engine.Connect.Search.Crawlers.AllCatalogItemsCrawler - Sitecore.Commerce.Engine.Connect.Search.Strategies.CatalogsIntervalAsynchronousStrategy - Sitecore.Commerce.Engine.Connect.Search.Strategies.CatalogSystemIntervalAsynchronousStrategyBase - Sitecore.Commerce.Engine.Connect.Search.Strategies.CategoriesIntervalAsynchronousStrategy - Sitecore.Commerce.Engine.Connect.Search.Strategies.SellableItemsIntervalAsynchronousStrategy - Sitecore.Commerce.Engine.Connect.Search.IndexUtility - Sitecore.Commerce.Engine.Connect.Search.Strategies.CommerceIntervalAsynchronousStrategy - Property: ItemsToTake.
Commerce Engine Connect Data Provider: - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.MappingEntries - Use MappingsByDeterministicId instead. - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.Bundles - Use MappingsByDeterministicId instead. - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.MappingEntriesLastUpdatedUtc - Use CheckMappings() instead. - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.UpdateMappingEntries - Use CheckMappings() instead. - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetCatalogName - Use GetCatalogMappingEntry(ID deterministicId) instead. - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetCatalogEntity - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.SetParentId - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetSitecoreIdForDeterministicId(string deterministicId) - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetPathIdsForSitecoreId(string sitecoreId, bool includeVariations = true) - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetPathIdsForSitecoreId(string sitecoreId, string parentId) - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetCatalogItemDeterministicIds - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetSitecoreIdFromMappings - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetEntityIdFromMappings - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetItemDefinition - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetListItemEntities - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetListItemSitecoreIds - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.GetListItems(string sitecoreId, string listName, string typeName, int take) - Sitecore.Commerce.Engine.Connect.DataProvider.CatalogRepository.DoUxAction
Commerce Engine Connect pipelines:

Removed as of Experience Commerce 10.0
- Sitecore.Commerce.Engine.Connect.Pipelines.Carts.AddCartLine - Sitecore.Commerce.Engine.Connect.Pipelines.Carts.UpdateCartLine - Sitecore.Commerce.Engine.Connect.Pipelines.Carts.RemoveCartLine

[end of document]