COMMUNICATIONS

MEDIA & ENTERTAINMENT

ENERGY & UTILITIES

INSURANCE

HEALTH

GOVERNMENT

DIGITAL INTERACTION PLATFORM

CME System Administration

Vlocity™

Table of Contents

System Administration for Vlocity Communications, Media, and Energy

Application Constraints

You can configure your Vlocity applications to remain within the operating limits or constraints of each application.

Module	Limitations
Salesforce Platform	Vlocity applications deployed to the Salesforce platform are subject to Salesforce limitations. Refer to Salesforce documentation for an enumeration of these limitations.

Module	Limitations
Configure Price Quote (CPQ)	

Non-Cacheable APIs

Vlocity provides a complete set of cart-based APIs for creating, updating and submitting carts such as Orders, Quotes, and Opportunities. These APIs are not cacheable because they perform CPQ calculations at runtime and update the state of the cart with each call.

Vlocity CPQ is fully hosted on the Salesforce platform. To avoid deployment problems and degradation in service quality, we recommend that the number of orders, quotes, order lines, and quote lines processed in your org not exceed the limits listed below. If your growing deployment exceeds the guidance shown in the following table, contact your Vlocity account representative to understand how Vlocity can scale to meet your needs.

Number of Line Items in Order*	Number of Products in Sales Catalog	Depth of Product Bundle Hierarchy per Line	Peak Number of API calls** per Hour	Peak Number of Orders*** per Hour
Up to 10	1,500	Up to 3	113,000	4,300
11 to 23	1,500	Up to 3	87,500	3,500

*The test was carried out with a catalog containing 1,500 products with an average of 12 attributes including 2 configurable attributes.

**The only load in the Salesforce Org was the Vlocity Cart APIs

***An order is comprised of, on average, 25 API calls during the shopping experience, from browsing to ordering.

The limits listed above are not hard limits. Vlocity tests CPQ to a higher level of throughput to ensure that unexpected high peaks are handled gracefully by the service. Your product model, number of pricing rules, number of compatibility rules, number of configuration rules, number of promotions, load, performance, and other system issues can prevent some limits from being reached. Stated limits are not a promise that the specified throughput is available in all circumstances.

Typical Shopping Funnel for Existing Customers

In this example, every new and returning customer places an order. The API calls in this example include:

- 15% New Customer Orders
- 85% Returning Customer Orders

Cacheable APIs

Vlocity also provides a set of cacheable APIs (also known as Digital Commerce APIs) that enable consumer shopping use cases where the user is often anonymous until well into the checkout process. These APIs cache responses by customer context in order to deliver faster response times and higher scalability than is possible using the cart-based non-cacheable APIs. These APIs also enable browsing for eligible offers without first creating a cart for the shopper.

Our CPQ Cacheable APIs are currently intended to support B2C Scenarios. While it is our intent to support various Product Catalog scenarios, it is important to note that specific guardrails must be respected to get the best performance out of these Cacheable APIs.

We recommend that the number of Baskets, Orders, and Order lines processed in your org not exceed the limits listed below. Contact your Vlocity account representative to understand how Vlocity can scale to meet your needs.

Typical Shopping Funnel for New Prospect

In this example, very little of the shopping traffic results in an order. The API calls in this example simulate the following shopping funnel:

- 75% Browsing Only
- 20% Browsing and Adding Items to Basket
- 05% Browsing, Adding Items to Basket and CheckOut

Module	Limitations

Number of Line Items in Baskets	Number of Products in Sales Catalog	Depth of Product Bundle Hierarchy per Line	Peak Number of API Calls per Hour (w/o Digital Commerce Gateway)	Peak Number of API Calls per Hour (w/ Digital Commerce Gateway)	Peak Number of Orders per Hour (w/o Digital Commerce Gateway)	Peak Number of Orders per Hour (w Digital Commerce Gateway)
Up to 10	1500	Up to 3	782,000	2,084,000	2,600	6,700
Up to 23	1500	Up to 3	602,000	1,602,000	1,900	5,000

The test was carried out with a catalog containing 1500 products with an average of 12 attributes including 2 configurable attributes per product.

The test warmed up all caches ahead of time so there are zero cache misses.

Typical Shopping Funnel for Existing Customers

In this example, every returning customer places an order, while many new prospects leave before checkout. The API calls in this example simulate the following shopping funnel:

- New Customer Orders
  - 20% Browsing Only
  - 10% Browsing and Adding Items to Basket
  - 05% Browsing, Adding Items to Basket and CheckOut
- Returning Customer Orders
  - 65% Browsing, Adding Assets and Items to Basket and CheckOut

Number of Line Items in Baskets	Number of Products in Sales Catalog	Depth of Product Bundle Hierarchy per Line	Peak Number of API Calls per Hour (w/o Digital Commerce Gateway)	Peak Number of API Calls per Hour (w/ Digital Commerce Gateway)	Peak Number of Orders per Hour (w/o Digital Commerce Gateway)	Peak Number of Orders per Hour (w/ Digital Commerce Gateway)
Up to 10	1,500	Up to 3	386,000	490,000	16,800	34,300
Up to 23	1,500	Up to 3	157,000	323,000	11,200	22,700

The test was carried out with a catalog containing 1500 products with an average of 12 attributes including 2 configurable attributes per product.

The test warmed up all caches ahead of time so there are zero cache misses.

Cache Miss Considerations

What is a Vlocity CPQ Cache Miss?

A Vlocity CPQ cache miss is a state in which the data requested for processing a CPQ API request is not found in the Vlocity API Response Cache. (If it were found, then it would be a cache hit.)

Cache misses lead to higher response times because the CPQ API must execute the CPQ engine to calculate the response, then it generates the response and stores it in the cache.

Frequency of Basket Cache Misses

The frequency of cache misses is determined by the basket shape. Several factors contribute to the basket shape, including those listed here:

- Number of Dimensions (which is determined by the eligibility rules)
- Number of configurable attributes
- Number of promotions applied per product
- Number of product line item combinations

Module	Limitations
	NOTE An increase in Basket shape permutations will result in an increase of cache misses. If this occurs, then we suggest that you leverage the CPQ Non-Cacheable APIs. **Duration of a Basket Cache Miss** There are several governing factors that affect the duration of a cache miss based on the data shape and the factors listed above. **Cache Warm-Up** Vlocity provides automated warm-up cache responses for anonymous browsing calls (GetOffers and GetOfferDetails). Several factors contribute to the performance of cache warm-up jobs, including those listed here: • The number of eligibility rules • The depth of the hierarchy of products • The number of offers in a sales catalog

Enterprise Product Catalog (EPC)	• Salesforce Security profiles are not supported.
	• You must manage product attributes using Product Designer or Product Console. Do not use Salesforce Classic.
	• The range for the number of Product Catalog translations in the Translation Object is up to 50,000.
	• The recommended levels for Product Hierarchy is 3 to 4.
	• Product Designer:
	  • Recommended assigned attributes per product: 50 or fewer.
	  • Objects, products, and offerings can have only one facet, which must be General Properties.

<table>
<tr><th>Module</th><th>Limitations</th></tr>
<tr><td>Order Management Standard</td><td>
Vlocity Order Management Standard is a fully Salesforce-hosted version of Vlocity Order Management. To avoid deployment problems and any degradation in service quality, we recommend that the number of orders processed in your org per hour not exceed the limits listed below. If your growing deployment needs to support more orders per hour, contact your Vlocity account representative to understand how Vlocity can scale to meet your demands.

All new Salesforce environments come with a limited set of High Volume Platform Events (HVPE) that improve the performance of Salesforce applications. Vlocity recommends that OM Standard use platform events to improve the overall performance of fulfillment. To configure High Volume Platform Events, see OM Standard: High Volume Platform Events (HVPE) for Orchestration in the Vlocity Order Management documentation on the Support Center.

Customers may need to purchase one or more HVPE Add On licenses to support more complex configurations.

The following table highlights the approximate performance of Vlocity Order Management Standard when using Platform Events:

<table>
<tr><th>Domain</th><th>Max Order Lines (Post-decomp)</th><th>Max Callouts / order (1.5 Sec Latency assumed)</th><th>Peak Number of Orders per Hour</th><th>Number of Orders per Day</th></tr>
<tr><td>Business to Consumer (B2C)</td><td>15</td><td>4</td><td>6,000</td><td>60,000</td></tr>
<tr><td>Business to Business (B2B)</td><td>75</td><td>20</td><td>1,200</td><td>12,000</td></tr>
</table>

For lower volumes of orders the traditional Salesforce batch processing mechanism can be used. In this situation the Platform Events setting must be turned off.

The following table highlights the approximate performance of Vlocity Order Management Standard when the traditional batch processing mechanism is used:

<table>
<tr><th>Domain</th><th>Max Order Lines (Post-decomp)</th><th>Max Callouts / order (1.5 Sec Latency assumed)</th><th>Peak Number of Orders per Hour</th><th>Number of Orders per Day</th></tr>
<tr><td>Business to Consumer (B2C)</td><td>15</td><td>4</td><td>800</td><td>8,000</td></tr>
<tr><td>Business to Business (B2B)</td><td>75</td><td>20</td><td>160</td><td>1,600</td></tr>
</table>

The limits listed in the above tables are not hard limits. Vlocity tests Order Management to higher levels of throughput to ensure that unexpectedly high peaks are handled gracefully by the service. However, stated limits may be lower as a result of product model complexity, system configuration, overall system load, or other system issues. OM Standard consumes from the shared resources in your Salesforce organization. Other processing may reduce your ability to achieve the stated limits. Customers must load test their configuration of OM with a realistic production background load to determine their estimated performance and scalability.

For Order Management Standard, customers are responsible for ensuring an adequate amount of Platform Events are available for appropriate operation of the Vlocity Order Management functions. For more information about HVPE, see Platform Events Developers Guide.
</td></tr>
</table>

<table>
<tr><th>Module</th><th>Limitations</th></tr>
<tr><td>Order Management Plus</td><td>Vlocity Order Management Plus leverages cloud-computing technology to provide the additional scalability that large telecommunications companies typically require. While the design time and runtime experience remain the same for most users, Vlocity Order Management Plus executes decomposition and orchestration functions outside of the Salesforce platform to provide the scalability required to support larger volumes of orders. Use Order Management Plus when exceeding the volumes listed in the OM Standard tables.

To avoid deployment problems and any degradation in service quality, we recommend that the number of orders processed in your org per hour not exceed the limits listed below. If your growing deployment needs to support more orders per hour, contact your Vlocity account representative to understand how Vlocity can scale to meet your demands.

The following table highlights the approximate performance of Vlocity Order Management Plus:

<table>
<tr><th>Domain</th><th>Max Order Lines (Post-decomp)</th><th>Max Callouts / order (1.5 Sec Latency assumed)</th><th>Peak Number of Orders per Hour</th><th>Number of Orders per Day</th></tr>
<tr><td>Business to Consumer (B2C)</td><td>15</td><td>4</td><td>15,000</td><td>100,000</td></tr>
<tr><td>Business to Business (B2B)</td><td>75</td><td>20</td><td>3,000</td><td>20,000</td></tr>
</table>

The limits listed above are not hard limits. Vlocity tests Order Management to higher levels of throughput to ensure that unexpectedly high peaks are handled gracefully by the service. However, stated limits may be lower as a result of product model complexity, system configuration, overall system load, and other system issues.

Completed orders older than one week must be purged from the RDS system (AWS) to maintain optimal database performance and reduce database storage costs.</td></tr>
<tr><td>Contract Lifecycle Management</td><td>Vlocity Web Templates have the following limitations:

• Vlocity provides a feature to generate a Microsoft Word file with .docx file extension from a HTML document. Only a subset of HTML tags are supported for conversion.
• Only a three-level hierarchy of embedded web document templates is reliably-supported, and this includes the top level parent template.
• Chart-generation and dynamic-insertion of images is not supported.
• In online-generation mode, documents with up to 200 line items can be reliably-generated, assuming each line item displays five columns.
• In batch-generation mode, documents with up to 3,000 line items can be reliably-generated, assuming each line item displays five columns.
• Batch-generation of a document only downloads the document to Microsoft Word format.

Vlocity for Microsoft Word Templates with .docx file extensions have the following limitations:

• Chart-generation and dynamic-insertion of images and hyperlinks is not supported.
• A maximum of 3,000 line items items can be reliably-generated using custom-class mapping documents.
• Does not support generation of documents in batch mode.</td></tr>
<tr><td>Retail</td><td>Retail Application Catalog supports at catalog with a maximum of two tiers.</td></tr>
</table>

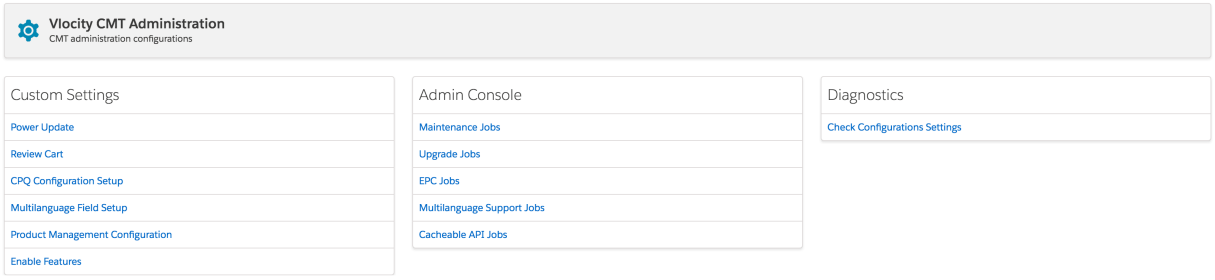
.

Setting Up Your CPQ Configuration Settings

You can configure many custom settings that control configure, price, quote (CPQ) features and behaviors.

To set up your CPQ configuration settings:

1. Go to the Vlocity CMT Administration tab.



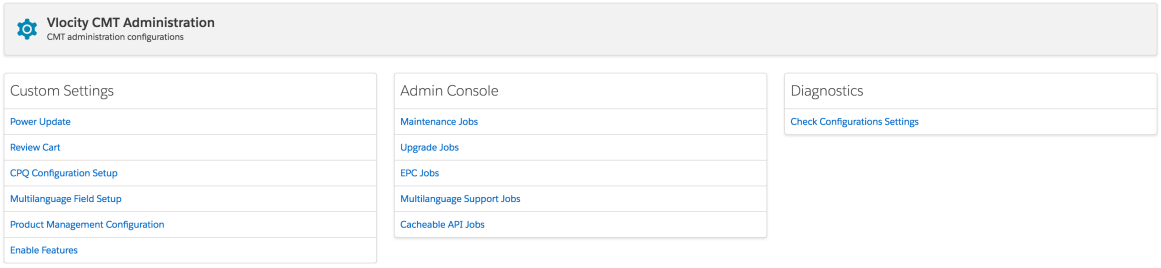
2. Under **Custom Settings** select **CPQ Configuration Setup**.
3. Click the **Action** button to modify a **Configuration Name** or **Setup Value**.
4. To view the current values for your Configuration Settings, return to the dashboard and under **Diagnostics**, select **Check Configurations Settings**.

Checking Your CPQ Configuration Settings

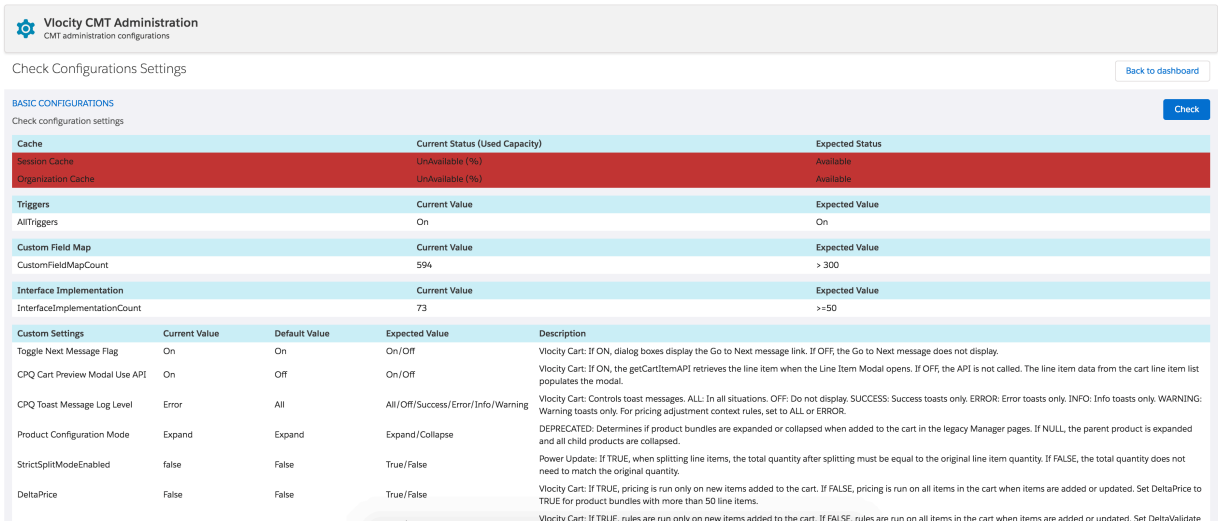
You can view the current values for your CPQ configuration settings.

To check your settings:

1. Go to the Vlocity CMT Administration tab:
   a. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
   b. Click **Vlocity CMT Administration**.



2. Under **Diagnostics**, select **Check Configurations Settings**.
3. Click **Check**.
   The current values for your Configuration Settings are displayed.



4. To modify settings, return to the dashboard, and, under **Custom Settings**, select **CPQ Configuration Setup**.

CPQ Configuration Settings Reference

Vlocity Communications, Media, and Energy includes these custom settings that control many configure, price, quote (CPQ) features and behaviors.

To work with CPQ configuration settings, go to the Vlocity CMT Administration tab, and under **Custom Settings** select **CPQ Configuration Setup**.

Active CPQ Configuration Settings

Setting Name	Description	Value
AddPromoValidate	Promotions: If set to true, runs compatibility (or validation) rules when promotions are added to the cart. If set to false, does not run compatibility rules promotions are added.	true/false Default: false
AllowOperationsOnDiscProducts	Vlocity Cart: If set to true, child products with a cardinality of 0,0,0 can be configured or deleted in a *MACD* order that is being updated. If set to false, line items with child products having a cardinality of 0,0,0 can't be deleted or configured.	true/false Default: false
AlwaysInsertMessageElement	If set to true, the API inserts the message element in all JSON nodes. If set to false or null, the API does not insert the message element. This must be set to true when using the persistent cart in an OmniScript. Used only with Cart-Based APIs.	true/false Default: true
AlwaysRegenerateV2AttributeFields	v2 attribute upgrades: If set to true, Line Items Attributes Conversion job regenerates all attributes. If set to false or NULL, the Line Items Attributes job only processes attributes not yet converted. To use this setting, set SaveProductAttributeMetadata to true.	true/false Default: true

Setting Name	Description	Value
ApplyPromoToBundleWithExpiredChildItem	If set to true, a promotion can apply to a bundle with past or end dated child items.	true/false Default: false
AttributeValueDisplaySortMode	Specifies sort order of attribute picklist values in API response based on the following values: • ID: Picklist values sort by ID • DISPLAYSEQUENCE: Picklist values sort by Display Sequence • VALUE: Picklist values sort alphabetically by Value Used only with Cart-Based APIs.	Default: VALUE
CacheAPI.BasketAPIPoolSize	When a the Basket API is called, this setting determines the number of Pseudo Orders that are created. The value should be an integer greater than 1.	Integer Default Value: 1000
CacheAPI.CacheManagement	Enables the Regenerate Cached API Records job, which will programmatically regenerate cached API responses when your product model or sales catalog has been modified. To enable this setting, go to **Setup** → **Develop** → **Custom Settings** → **Trigger Setup**. Add a new trigger setup detail and select the **Trigger On** checkbox for CacheAPI.CacheManagement. See Updating the API Cache.	on/off Default: off
CacheAPI.CombinationLimit	Limits the number of results that can be cached in a catalog by one EcomDataProfileGenerator cache job. Caching combinations that exceed the limit are cached in multiple jobs. Used only with Digital Commerce APIs.	Integer Default: 2000
CacheAPI.CreateCartBatchSize	When creating carts in bulk, the value of this setting determines whether the createCart API call is asynchronous or synchronous. If the number of baskets in the request is less than the value of this setting, the createCart API call is synchronous. If the number of baskets in the request is greater than or equal to the value of this setting, the createCart API call is asynchronous. For asynchronous createCartcreateCart calls, this setting indicates the number of baskets that will be processed per batch. Used only with Digital Commerce APIs.	Integer Default: 5
CacheAPI.CreateCartFromContextKey	If set to true, the Create Cart from Basket uses the cartContextKey passed in the input to create the cart. If set to false or null, the JSON result from the GetOfferDetails API must be passed in the request body to create the cart. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPI.CustomGroupContextItems	Specifies a comma-separated list of user-defined custom context types. Used only with Digital Commerce APIs.	String Default: empty string (null)
CacheAPI.ExternalSystemAuthHeader	Specifies the authentication header of the AWS (external) system to which the cache will be migrated from Salesforce via the CMT Administration Cacheable API Jobs page. See Initializing the API Cache.	String Default: empty string (null)
CacheAPI.ExternalSystemPartitionKey	Specifies to the system partition key of the AWS (external) system which the cache will be migrated from Salesforce via the CMT Administration Cacheable API Jobs page. See Initializing the API Cache.	String Default: empty string (null)

Setting Name	Description	Value
CacheAPI.ExternalSystemURL	Specifies the endpoint of the AWS (external) system to which the cache will be migrated from Salesforce via the CMT Administration Cacheable API Jobs page. See Initializing the API Cache.	String Default: empty string (null)
CacheAPI.GenericOffersAttribute	If set to true, the attribute category structure in getOffers API responses are changed to be consistent with the generic attribute category structure of other APIs such as getOfferDetails. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPI.GetOffersCacheRecLimit	Limits the number of results that can be cached in a catalog by one GetOffers or GetContainsOffers cache job. Must be ≤ CacheAPI.CombinationLimit Caching combinations that exceed the limit are cached in multiple jobs. Used only with Digital Commerce APIs.	Integer Default: 2000
CacheAPI.GetRelOffersCacheRecLimit	Limits the number of results that can be cached in a catalog by one ContextEligibilityGenerator cache job. Must be ≤ CacheAPI.CombinationLimit Caching combinations that exceed the limit are cached in multiple jobs. Used only with Digital Commerce APIs.	Integer Default: 2000

Setting Name	Description	Value
CacheAPI.IncludeParentContextPLEs	This parameter is supported for both anonymous and logged-in user scenarios. *When `CacheAPI.IncludeParentContextPLEs = True:`* For offer details of a product: • Returns all the adjustments and overrides in the getOfferDetails API for promotions, root, and child products. • The price result for the root product will include the base prices for the root product and all of the adjustments or overrides defined for the root product. The price result for child products will include the base prices for the child products and all of the adjustments or overrides defined for the child products. For offer details of a promotion: • The price result for promotion products will include the base prices for the promotion products and all of the adjustments or overrides defined for the promotion products. The price result for child products of promotion products will include the base prices for the child products and all adjustments or overrides defined for the child products. *When `CacheAPI.IncludeParentContextPLEs = False`* For offer details of a product: • The price result for the root product will include the base prices for the root product. The price result for child products will include the base prices for the child products and all of the adjustments or overrides defined for that product under the root product. For offer details of a promotion: • The price result for promotion products will include the base prices for the promotion products and the adjustments or overrides defined for that promotion product for that promotion. The price result response for the child products of promotion products will include base prices for the child products and the adjustments or overrides defined for that promotion and the adjustments or overrides defined for that promotion product. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPI.MaxCombinationLimit	Limits the total number of data profile results that can be cached in each catalog. Used only with Digital Commerce APIs.	Integer Default: 100000
CacheAPI.MaxGetOffersCacheRecLimit	Limits the total number of offer results that can be cached in each catalog. Must be ≤ CacheAPI.MaxCombinationLimit Used only with Digital Commerce APIs.	Integer Default: 100000
CacheAPI.MaxGetRelOffersCacheRecLimit	Limits total number of context eligibility results that can be cached in each catalog. Must be ≤ CacheAPI.MaxCombinationLimit Used only with Digital Commerce APIs.	Integer Default: 100000

Setting Name	Description	Value
CacheAPI.OptimizedBasketResponse	The following values determine the response format during a remote call: • true: Basket API responses are serialized in a remote call. • false: Remote call responses have a mapped format. This setting has no impact on REST responses. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPI.PerfMode	Improves the cache-miss performance of the Add To Basket API by skipping certain logic. Refer to the following list of values to determine the impact of each setting: • AT - Reduces the number of attributes returned in the attributeCategories node of the basket response. • RAM - By default basket APIs evaluate whether required attributes are missing in the basket configuration. With this setting, the evaluation is not performed; therefore, errors in the basket response are not reported even if a required attribute is not configured. • DMA - Because discounts and manual adjustments are not supported in Digital Commerce APIs, this setting skips those steps in the pricing plan. • DCF - Does not return back-end or technical fields in the basket response. • DT - Disables the Telco Account and Telco Order OOTB trigger for pseudo record creation. • BC - By default, the basket APIs evaluate whether basket cardinality is violated in the basket configuration. Using this setting, basket cardinality is not checked. Therefore, cardinality errors are not returned in the basket response even if one or more line items violate cardinality.	String Default Value: null Values: AT, RAM, DMA, DCF,
CacheAPI.RegenerateCatalogBatches	Controls the populate cache batch jobs that run when the Regenerate Cached API Records job is run. The default value is `ContextEligibilityGenerator,GetOffers,GetContainOffers`. If you do not want to regenerate entries for the Get Contains Offers API, set the value to `ContextEligibilityGenerator,GetOffers`.	ContextEligibilityGenerator,G
CacheAPI.RegenerateEntriesPartition	Sets the maximum number of flex queue jobs that can be submitted by the system at one time during the regenerate batch job run.	Integer Default 90
CacheAPI.RegenerateFutureDatedCache	If `CacheAPI.RegenerateFutureDatedCache=true`, running the Regenerate Cached API Records batch job pops up a window that allows you to set the date and time at which the cached records become effective. Setting this parameter to false does not give you the option to enter a date and time and makes the cache effective when the cache jobs complete. See: Updating the API Cache.	
CacheAPI.SkinnyBasket	When CacheAPI.SkinnyBasket is set to true, AssetToBasket just returns key-value pairs, which can improve performance. When CacheAPI.SkinnyBasket is set to false, AssetToBasket provides the full basket response (backwards compatible).	true/false Default: false
CacheAPI.SkinnyCtxDmsnCombinations	If CacheAPI.SkinnyCtxDmsnCombinations is set to true, context dimension combinations are generated based on context rules. If set to false, all possible context dimension combinations are generated.	true/false Default: false
CacheAPI.TimeToLiveInDays	Specifies the number of days an API cache response is effective after its start datetime. Used only with Digital Commerce APIs.	Integer Default: 30

Setting Name	Description	Value
CacheAPI.Trimmode	If set to true, the Basket APIs return a basket that has field and attribute metadata removed from the Basket API response, which reduces JSON processing and network I/O. If set to false, the Basket APIs return the basket as in previous releases. This setting overrides the `CacheAPI.SkinnyBasket` configuration setting.	true/false Default: false
CacheAPIFields	If set to true, the APIs cache responses. If set to false or null, the APIs do not cache responses. To use this setting, you must have a Vlocity Communications APIs Caching license. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPIPartialConfigure	Set this value to true to split the Get Offer Details API into two transactions. For CME Summer '19 and later, this setting is not used if MTSEnabled is set to true. See GetOfferDetails API. Used only with Digital Commerce APIs.	true/false Default: false
CacheAPIPartialConfigureProduct	If set to true, the configure offer and addToBasket APIs split into two transactions, allowing two-step product configuration. If false or null, these APIs operate in a single transaction. For CME Summer '19 and later, this setting is not used if MTSEnabled is set to true. Used only with Digital Commerce APIs.	true/false Default: false
CachedQueryMode	Vlocity caching: If set to true, uses output from queries in the CacheQueryStore. If set to false, queries directly from the database. Default is false.	true/false Default: false
CacheEnabled	Supports platform caching. To use context rules, including the Tightest Match service, CacheEnabled must be set to true. See Enable the CPQ Cache.	true/false Default: true
Cart Level Discount Approval Required	Vlocity Cart: If set to true, cart-added discounts must be approved and are created with an ApprovalStatus field value of Not Submitted. If set to false, no approval is required and the ApprovalStatus field will be set to Approved. To implement your own approval system, set Cart Level Discount Approval Required to true and configure the CpqCustomDiscountApprovalService class in your org. See Approving Discounts and DefaultApprovalStatusImplementation.	true/false Default: false
ContextRulesCacheMode	The following ContextRulesCacheMode values control the caching of data related to context rules: • `cachemode`: All the data and wrapper collections are cached using platform cache. • `inmemmode`: Data and wrapper collections are not cached and will instead remain in memory.	Default: `inmemmode`
ContextRulesEnabled	Context Rules, including the Tightest Match service: If set to true, context rules are enabled for the org. If set to false, context rules are disabled.	true/false Default: false

Setting Name	Description	Value
ControlProduct2FieldsDuringCreation	Vlocity EPC: Set this to true if you created a large number of custom fields in Product2 and have slow performance issues when you create a product. If true, Product2 lookup fields populate only the Name field and do not populate entire fields during *SOQL* queries.	true/false Default: false
CPQ Cart Preview Modal Use API	Vlocity Cart: If set to on, the getCartItemAPI retrieves the line item when the Line Item Modal opens. If set to off, the API is not called. The line item data from the cart line item list populates the modal.	on/off Default: on
CPQ Toast Message Log Level	Vlocity Cart: Controls toast messages using the following values • ALL: In all situations • OFF: Do not display • SUCCESS: Success toasts only • ERROR: Error toasts only • INFO: Info toasts only • WARNING: Warning toasts only For pricing adjustment context rules, set to ALL or ERROR.	Default: ALL
CustomRuleMessageFieldName	Advanced rules: Specifies the API field name on the product relationship object that is used to store a custom message for each product relationship.	Field name, such as Message
DefaultEntityFilterEditFieldset	Advanced rules: Specifies the field set name for the entity filter pages, which enables you to edit entity filters. Note: If you are upgrading from v15.2 or prior, ensure that this setting is present and includes the value Entity_Filter_Edit_Field_Set (the default).	String
DefaultHierarchy	Specifies the default product hierarchy level to return. To return all levels, do not use this custom setting. When this custom setting is removed, search works faster. Used only with Cart-Based APIs.	Integer greater than 0, such a
DefaultPowerUpdateJobThreshold	Power Update: Specifies a threshold of line items above which Power Update spawns an *Apex* job.	Integer Default: 200
DefaultPricingPlan	Pricing Plan Service: Specifies the code of the default pricing plan used for the org. Used only when PricingPlanService is the active implementation for the Pricing Interface. If using the out-of-the-box plan, enter DEFAULT_PRICING_PLAN (the default).	String
DefaultSplitJobThreshold	Power Update: Sets the threshold of splits above which Power Update spawns an *Apex* job to split individual copies of a line item.	Integer
DeleteServices	Defines how delete services behave for promotions. `No` sets shallow delete mode, which deletes only discounts, not products associated with promotions. `Yes` sets deep delete mode, which deletes discounts and products and services that are associated with promotions.	Yes/No Default: Yes
DeltaPrice	Vlocity Cart: If set to true, pricing is run only on new items added to the cart. If set to false, pricing is run on all items in the cart when items are added or updated. Set DeltaPrice to true for product bundles with more than 50 line items. This setting is not supported by Digital Commerce APIs. See Digital Commerce API Usage Considerations.	true/false Default: false

Setting Name	Description	Value
DeltaValidate	Vlocity Cart: If set to true, rules are run only on new items added to the cart. If set to false, rules are run on all items in the cart when items are added or updated. Set DeltaValidate to true for product bundles with more than 50 line items. This setting is not supported by Digital Commerce APIs. See Digital Commerce API Usage Considerations.	true/false Default: false
EnableMultiServiceBulkAction	If the EnableMultiServiceBulkAction multi-service setting is set to true, the following buttons are displayed in the Group Management UI: • Apply To Group • Create and Submit Order(s) • Validate and Price If set to false (the default), these buttons are not displayed.	true/false Default: false
EnableSupplementalOrderPricing	If set to true, pricing will be invoked based on the price flag in the API request. If set to false (the default), pricing will not be invoked when creating a supplemental order. Used only with Cart-Based APIs.	true/false Default: false
FDOStatuses	This setting helps configure the filter criteria in queries or code related to merging line items with future dated orders (*FDO*). The default value is: `Status=Draft;OrderStatus__c=Ready To Submit,Queued` Do not modify this setting's default unless you need to configure the two order fields that are queried: Status and OrderStatus__c. In asset-to-order *MACD* FDO flow, you create orders on a future date with the selected assets already added to the Order as OrderItems. By default, the system looks for OrderItems in Orders that are in Status = Draft AND OrderStatus__c is either Ready to Submit or Queued. To add a new OrderStatus__c value, override the filter this way: OrderStatus__c=Ready to Submit,Queued,In Progress In this case, the values specified here override the default OrderStatus__c values. To add a new value to Status, set the FDOStatuses setting value to: Status=Draft,Activated In this case, only the values for Status are overridden. Values for OrderStatus__c remain as default values.	String Multiple fields use a semi-col expressions. If there is only one field, the s Multiple values for a field are
FilterRuleWithEligAvail	Checks the availability and eligibility of the productToBeAdded by rule. Determines if eligibility and availability are run on a product when the product is added to the cart.	true/false
FindReferencesRowCountLimit	Vlocity *EPC*: Limits the number of rows returned across products, promotions, quotes, orders, opportunities and assets using the Find References button in *Vlocity Product Console*. The maximum is 1000. If null, the default limit is 20 rows.	Integer Default: 20

Setting Name	Description	Value
GuestHandlingEnabled	When set to false, GuestHandlingEnabled disables the secure guest user feature. By default, its value is true without explicitly specifying an entry (recommended). You only need to specify this setting if you need to disable the guest user feature. See Guest User Security Restrictions.	true/false Default: true
IncludeAddOnsAttributes	Can be used to check attributes before adding the associated child product to the cart. When IncludeAddOnsAttributes is set to true, the API returns attributes for a product and all its child products (add ons) in the product hierarchy.	true/false Default: false
LevelBasedApproach	If set to true, APIs return root and first-level children in product hierarchy. Next level are returned when user clicks expand icon on product. If set to false, APIs return full product hierarchy. To use this setting, CacheEnabled must be set to true. Used only with Cart-Based APIs.	true/false Default: false
LevelBasedPageSize	Specifies the number of children to include for getCartsItems and addToCart APIs calls. To use this setting, you must set CacheEnabled and LevelBasedPagination to true. See Configuring Cart Item Refresh Levels. Used only with Cart-Based APIs.	Integer Default: 10
LevelBasedPagination	If set to true, APIs limit children in response to LevelBasedPageSize setting value. If LevelBasedPageSize is null, the default is 10. If set to false, APIs return full product hierarchy. To use this setting, CacheEnabled must be set to true. Used only with Cart-Based APIs.	true/false
LockOrderAfterSubmitToOM	Specifies whether the order is locked for editing. • true: If the order or order item not in Draft status, it is locked for editing. • false: The order or order item can be edited. This is was behavior prior to CME Winter '20. Used only with Cart-Based APIs.	true/false Default: false
LogOdinNotifications	Specifies one of the following values to control when Order Event Log Entry records are created: • DoNotInsert: Disable notification logging. • InADifferentTransaction: Insert notifications in OrderEventLogEntry__b in a different transaction. • InSameTransaction: Insert notifications in OrderEventLogEntry__b in the same transaction. This setting controls only the logging of in-flight order notifications. The actual processing of notifications is independent of this setting. Any transaction-level OdinService.orderEventLogMode values override the value of LogOdinNotifications for those transactions unless set to null.	Default: DoNotInsert
MarginCalculationType	Setting for Cost and Margin. • margin = (price-cost)/price • markup = (price-cost)/cost Add this setting manually.	margin/markup Default: markup

Setting Name	Description	Value
MaxCPUTimePerTransaction	If MTSEnabled is set to true, this value sets the maximum allowed CPU time per *Apex* transaction before the transaction is split. Used only with Digital Commerce APIs.	Integer Default: 9500
MaxHeapSizePerTransaction	If MTSEnabled is set to true, this value sets the maximum heap size per Apex transaction before the transaction is split. Used only with Digital Commerce APIs.	Integer Default: 5000000
MTSEnabled	When MTSEnabled is set to true, the Multi-Transaction Service splits Apex transactions into multiple transactions to avoid exceeding Salesforce Governor limits. Used only with Digital Commerce APIs.	true/false Default: true
Multiple Promotion Nodes Resolution	Promotions: Use this setting in custom applications to select line items and apply promotions to the line item. If set to true, node IDs must be supplied for the promotion. If set to true or null, promotions are applied in the sequence defined in the promotion.	true/false (or null) Default: false
MultiServiceSingleOrderRecordTypes	Specifies a comma-separated list of Order record type Developer Names to make the custom record types eligible to be used in a multi-service single-cart flow. For example: `SalesOrder,BusinessOrder`	String Default: EnterpriseOrder
MultiServiceSingleQuoteRecordTypes	Specifies a comma-separated list of Enterprise Quote record type Developer Names to make the custom record types eligible to be used in a multi-service single-quote flow. For example: `EnterpriseQuote,BusinessQuote`	String Default: EnterpriseQuote
NumberOfDMLPerTransaction	If MTSEnabled is set to true, this value sets the maximum allowed data manipulation language (DML) statements per Apex transaction before the transaction is split. Used only with Digital Commerce APIs.	Integer Default: 125
NumberOfQueriesPerTransaction	If MTSEnabled is set to true and the current *SOQL* count exceeds the value of this setting, the transaction is split. Used only with Digital Commerce APIs.	Integer Default: 60
NeedContextForAttribConfig	Advanced rules and Vlocity Cart-Based APIs: If set to true, executes queries and code related to context checking while running Advanced Rules of type Attributes Configuration in the cart. If set to false, these queries, code and rules are not run, which may improve performance.	true/false Default: false
ObjectCopierPrice	Quote-to-order: If set to false, does not rerun the pricing for the order.	true/false Default: true
ObjectCopierValidate	If set to false, turns off validation when running DefaultObjectCopierImplementation Quote-to-order: If set to false, does not rerun the validation of the order.	true/false Default: true

Setting Name	Description	Value
OrderItemIgnoredColumns	Specifies a comma-separated list of OrderItem columns to be ignored in the configure and AddToBasket APIs. For example, specify `vlocity_cmt__AssetReferenceId__c` so that asset reference IDs are ignored when creating order items from the offer details. Used only with Digital Commerce APIs.	String Default: null
OriginalOrderCancellationStatusChanges	When set to off, canceling the original order will set the: • Order status to Superseded • Order fulfillment status to Superseded • Supplemental order status to Canceled • Supplemental order fulfillment status to Activated When set to on, users can cancel the original order while it is in flight, which will set the: • Order status to Canceled • Order fulfillment status to Superseded • Supplemental order status to Completed • Supplemental order fulfillment status to Activated This setting is only applicable to the cancellation of original orders. It is not considered for amended or supplemental orders.	on/off Default: on
PopulateAllActionVariants	Vlocity Cart-Based APIs in Winter '18 only: If set to true, populates both VFRemote and REST actions in response JSON, to ensure backward compatibility. If set to false or null, populates only VFRemote or REST sections in response actions, matching the request protocol, which increases performance.	true/false Default: false
PriceBookRefreshBatchSize	Vlocity EPC: Limits the size of attributes and products processed in the Refresh Pricebook batch job to remain within the Salesforce governor limits.	Integer Default: 200
PricingElementServiceLogging	Enables pricing element logging for debugging. Logging can consume high amounts of CPU. To prevent a heap size error, always disable this option for production environments.	true/false
PricingPlanServiceLogging	Enables pricing plan logging for debugging. Logging can consume high amounts of CPU. To prevent a heap size error, always disable this option for production environments.	true/false
ProductAttributesBatchProcessorSize	Specifies the size of the product attributes processor batch job. Depending on the number of attributes in a product, you can configure this value to avoid exceeding Salesforce governor limits.	Integer Default: 500
ProductHierarchyBatchProcessorSize	Specifies the size of the product hierarchy processor batch job. Depending on the product's hierarchy, you can configure this value to avoid exceeding Salesforce governor limits.	Integer Default: 20
RollupActionCodeField	Pricing: Specifies the field name of the cart's custom line item field that stores the Change action when a price is rolled up from a child product to its parent during an Asset-to-Order transition. If null (the default), no Change action occurs.	String Default: null
SaveProductAttributeMetadata	Vlocity EPC: Enables v2 product attribute metadata. If set to true or null, v2 attribute metadata is stored in AttributeDefaultValues__c and AttributeMetadata__c in shared catalog. If set to false, v2 attribute metadata is not stored on products in shared catalog.	true/false Default: true

Setting Name	Description	Value
ShouldConsiderDeletedItems	Advanced Rules: If set to true, rules evaluate deleted line items in the cart when applying product relationship rules such as auto-add, auto-remove, recommends, requires, and excludes. If set to false or null, rules disregard line items that are deleted in the cart.	true/false Default: false
SimplePLERuleCacheMode	Context Rules, Tightest Match: If set to true, price list entries with context rules are cached in the CPQPartition org cache for performance. If set to false, price list entry eligibility is checked using any context rules. Do not use if context rules use functions.	true/false Default: false
SingleFdoMode	Vlocity FDO: If set to true, extra queries or code related to merging line items with future dated orders will NOT execute because only a single future dated order is created per asset. This may improve performance. If set to false, such queries or code will execute.	true/false Default: false
StrictSplitModeEnabled	When splitting line items, determines whether to ensure the total quantity is equal to the original number.	true/false
Toggle Next Message Flag	Vlocity Cart: If ON, dialog boxes display the Go to Next message link. If OFF, the Go to Next message does not display.	on/off Default: on

Setting Name	Description	Value

UOWMode	If set to true, enables Unit of Work performance enhancements that save data manipulation language (DML) and Salesforce Object Query Language (SOQL) statements when making Cart-Based API calls. If UOWMode is set to false, these performance enhancements are not enabled.
	
	WARNING
	• Contact your Vlocity account representative before modifying the value of this setting.
	• Custom validation and pricing implementations may require modifications.
	• Digital Commerce APIs do not yet support a UOWMode setting of true.
	
	When UOWMode is enabled, the following features are supported:
	
	• **No Response Mode**: Returns only a success or failure message from the API instead of a full response. This saves time and improves performance. This feature supports Multi-add.
	• **Multi-add**, which supports:
	  • Root item multi - add: You can add multiple offers or Bundles in a single API call by passing the different root Pricebookentry Ids as comma-separated values to itemId.
	  • Child item multi – add: You can add multiple child items under a common parent in a single API call by passing the different child's Pricebookentry Ids as comma-separated values to itemId.
	• **Multi-delete**, which supports:
	  • Root item multi – delete: You can delete multiple offers or bundles in a single API call by passing the different root Line Item Ids as comma-separated values to itemId.
	  • Child item multi - add: You can delete multiple child items under a common parent in a single API call by passing the different child's Line Item Ids as comma-separated values to itemId.
		true/false
		
		Default: false

Setting Name	Description	Value
UseAssetReferenceIdForParentAndRoot	If set to true, enables performance enhancements that save data manipulation language (DML) statements in Add-to-Cart and other transactions. Before setting UseAssetReferenceIdForParentAndRoot to true, add a new field to Field Mapper for the Order Product → Asset objects: 1. Select Order Product as the source object and Asset as the destination object. 2. Map OrderItem RootItemId__c to Asset RootItemId__c. 3. Map OrderItem ParentItemId__c to Asset ParentItemId__c. **WARNING** • Contact your Vlocity account representative before modifying the value of this setting. • Setting UseAssetReferenceIdForParentAndRoot to true and then back to false can cause data corruption. • Digital Commerce APIs do not yet support a UseAssetReferenceIdForParentAndRoot setting of true.	true/false Default: false
UseWeights	Context Rules and Tightest Match: If set to true, weights from context dimensions are considered when determining the tightest match price list entry. If set to false, weights are ignored. Weights are always ignored if SimplePLERuleCacheMode is true.	true/false Default: false
UsingPromotions	Promotions and Vlocity Cart APIs: If set to true, executes queries and code processing related to promotion operations in the cart (not in the Promotions list). If set to false, does not execute such queries and code, which may improve performance.	true/false Default: true
UsingWeights	Context Rules and Tightest Match: If set to true, weights from context dimensions are considered when determining the tightest match price list entry. If set to false, weights are ignored. Weights are always ignored if SimplePLERuleCacheMode is true.	true/false Default: false
ValidateAndUpgradeV2Attributes	v2 Attribute Upgrades: If set to true, APIs upgrade attributes in the cart before submitting. If set to false or null, the cart functions as usual. Only use while a large upgrade job is still running. To use this setting, set SaveProductAttributeMetadata to true.	true/false Default: false

Deprecated CPQ Configuration Settings

The Opportunity, Order, and Quote Managers are deprecated. Use Vlocity Cart.

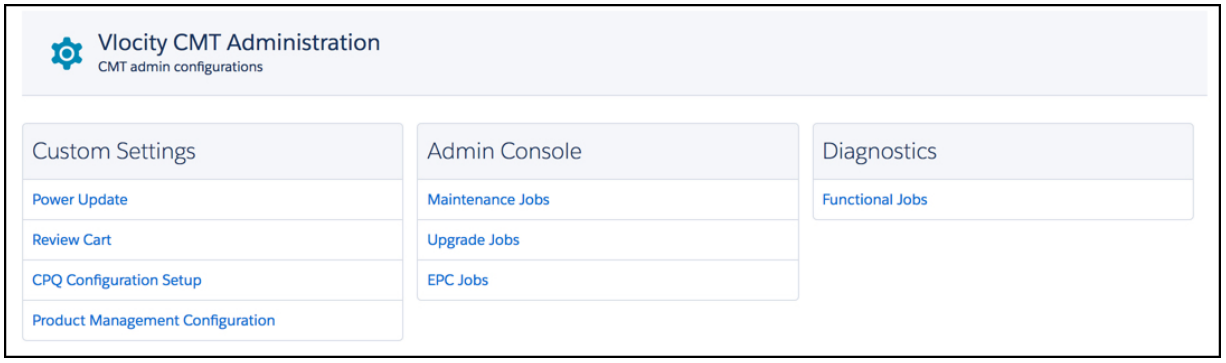
Deprecated Setting Name	Purpose	Values	Versions
Application.Opportunity.ShowSummary Application.Order.ShowSummary Application.Quote.ShowSummary	DEPRECATED: If set to true, the Order Manager displays the header summary in the legacy Order/ Quote/Oppty Manager pages. If set to false, no header summary will display.	true/false	
Attribute Check Mode	DEPRECATED: Controls the AddToCart button in the legacy Order/Quote/Oppty Manager pages. When set to true, if any product is missing a required attribute value and the product quantity is greater than zero, the Add to Cart button is disabled. When set to false, the Add to Cart button is present. However, the Create Quote, Create Order, or Create Asset buttons are disabled.	true/false Default: true	CME 11.43 and later
CacheAPI.CheckBasketEligibility	Determine whether the Digital Commerce API checks the cartContextKey parameter for the eligibility of the user's basket: • true: Check for the eligibility of the basket. • false: The API does not check for the eligibility of the basket. Used only with Digital Commerce APIs. With CME Fall '20 and later releases, use the basket lightweight validation feature instead.	true/false Default: false	CME Summer '19 and later. Deprecated with CME Fall '20 .
Cart Customizable Attribute Level	DEPRECATED: Applicable to Vlocity Cart in Summer '17. If set to ATTRIBUTE, Vlocity uses the legacy attribute metadata structure. If set to PRODUCT ATTRIBUTE or null, Vlocity Cart uses new attribute metadata structure. Not needed for new installs.	Attribute/ Product Attribute or null	Summer '17
CPPreProcOpportunityDRBundle CPPreProcOrderDRBundle CPPreProcQuoteDRBundle	DEPRECATED: Used in calculation procedures to specify the name of the DataRaptor used in the CalcProcPreProcessor for attribute-based pricing using PricingRulesImplementation. This functionality is replaced by attribute-based pricing available on Vlocity Process Library.	String	
CPPreProcUseDisplayText	DEPRECATED: Used in calculation procedures to specify whether to use the display value or actual value of an attribute in attribute-based pricing using PricingRulesImplementation. This functionality is replaced by attribute-based pricing available on Vlocity Process Library.	true/false	
CpqPerfMode	DEPRECATED: If set to true, validate and refresh price icons appear next to the preview in in the legacy manager pages. AddtoCart will not price or validate in Perf Mode. If set to false, the icons will not display.	true/false	CME 13 and later
CPQSubmit	DEPRECATED: When the order is submitted, invokes the work flow-based implementation. This functionality is replaced by CPQ/SubmitOrder OmniScript. See CPQSubmitFlowImplementation.	Flow	CME 10 and later

Deprecated Setting Name	Purpose	Values	Versions
CPQSubmitOrder	DEPRECATED: When the order is submitted, invokes the order management-based implementation. This functionality is replaced by CPQ/SubmitOrder OmniScript. See CPQSubmitFlowImplementation.	Flow	CME 12 and later
DefaultAccountFieldset	DEPRECATED: Specifies the API name of the field set to show fields in the custom account lookup dialog box from the review cart in the legacy Manager pages.	API name	
DefaultRenumberThreshold	DEPRECATED: Sets the threshold above which the cart items are renumbered so there is no gap in the line numbers in the legacy Manager pages.	Integer	CME 11 and later
ImplicitPricing	DEPRECATED: Determines if the pricing service runs every time a change to the cart is made in the legacy Order/Quote/Oppty Manager pages. If set to true, pricing occurs only if one or more of the fields have changed. If set to false, pricing does not occur when fields change.	true/false	CME 12–14 only
PaginationEnabled	DEPRECATED: If set to true, getProducts API returns only a set of products to be shown on the first page in the legacy Order/Quote/Oppty Manager pages. Next set of products can be traversed by clicking on Next or Previous link. If set to false, getProducts API returns the complete list of products that are qualified.	true/false	
PaginationSize	DEPRECATED: Specifies the number of product records getProducts API should return in the legacy Order/Quote/Oppty Manager pages. Used with PaginationEnabled.	Integer Default: 100	
Product Configuration Mode	DEPRECATED: Determines if product bundles are expanded or collapsed when added to the cart in the legacy Order/Quote/Oppty Manager pages. If the Product Configuration Mode setting is missing or has no value, the parent product is expanded and all child products are collapsed.	Expand/ Collapse	CME 11.4 and later
Run Trigger And WorkFlow	DEPRECATED		CME 14.1 and later
UseSessionCache	DEPRECATED	false	CME Spring '17 and later

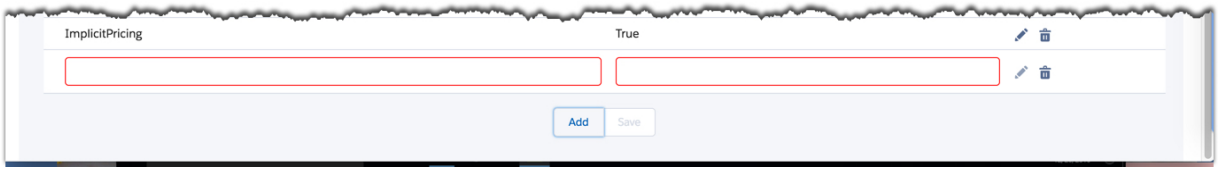
Adding CPQ Configuration Setup Custom Settings

You can add custom settings on the CMT Administration tab's CPQ Configuration Setup page.

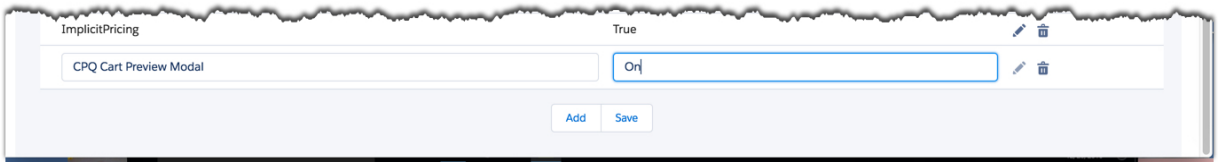
1. Go to the Vlocity CMT Administration tab, and under **Custom Settings** select **CPQ Configuration Setup**.



2. At the bottom of the CPQ Setup Configuration settings page, click **Add**.
   Two red fields appear.



3. In the field on the left, enter the setting name.
4. In the field on the right, enter the desired value.



5. Click **Save**.

Configuring CPQ Platform Cache

As part of the managed package, your org supports the platform cache, specifically the org cache, so you can use CPQ. Your org has a partition for the platform cache named CPQPartition. If the CPQPartition does not exist, you can create a default partition and name it CPQPartition.

You are allocated a minimum of 10 MB cache space to distribute on your org. Allocating a value of 1 MB to each partition allows you to use all of the features of CME.

After applying Vlocity Communications, Media, and Energy package upgrades, the vlocity_cmt cache partitions are always reset to zero. After you upgrade, allocate adequate space for the vlocity_cmt cache partition, CPQPartition.

Using the cache improves performance and response time for some operations, especially in large, deep hierarchical product bundles.

Vlocity uses the platform cache for the following items:

- Caching the complete hierarchy for product bundles in the shared catalog.

When you run the Refresh Platform Cache job on the Vlocity CMT Administration tab, Vlocity copies the shared catalog's complete hierarchy, including the details of all product child items, and caches that information in the org level cache layer. If you don't run this job, Vlocity caches the hierarchy at run time when a product is used for the first time. This job also updates stored filterable attributes.
When you add or modify any product bundle or product child item, you must run the Product Hierarchy Maintenance job to create a flat data store of the complete product hierarchy. Then run Refresh Platform Cache to copy the hierarchy to the platform cache and invalidate the older cache values.

- Caching the complete hierarchy for all products in a particular price book.
  When you click **Refresh Pricebook** on the Price Book record detail page, Vlocity calculates the complete hierarchy, including the details of all product child items, and caches that information in the org level cache layer. If you don't click **Refresh Pricebook**, Vlocity caches the hierarchy at run time when a product is used for the first time. Refreshing the price book also updates stored filterable attributes. If the **Refresh Pricebook** button is not visible, you must expose it. See Expose the Refresh Pricebook Button

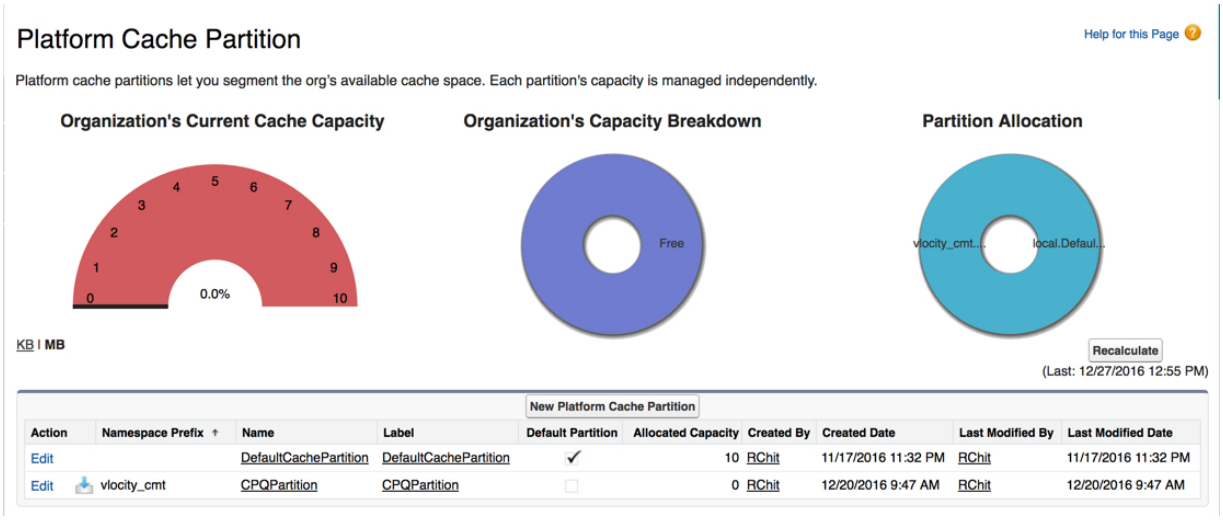
> **NOTE**
> When you add or modify any promotion, product, or product child item, refresh the price book to invalidate the older cache values and cache the updated values. For more information, see Refreshing the Price Book Prior to Winter '18.

- Caching product IDs to add by default when the product is selected.
  Vlocity caches the product IDs for all of the products with a quantity greater than zero in a bundle. This applies to each root product available.
- Caching the complete map with the key and values as the product wrapper for an entire bundle.
  The map contains the complete product hierarchy with the bundle root ID as the key. Vlocity generates a unique key value mapping all products wrapped in a bundle using the crypto class by passing the product hierarchy path as an argument. There is one map cached for each bundle in an entire price book.
- Caching a list of all product and promotion IDs.
  Vlocity caches a complete list of all product IDs in a particular bundle. Instead of iterating over the product bundles map, uses this list directly whenever a process requires a complete list of product IDs in a bundle. Use this when merging product child items across line items.

In addition, Vlocity Communications, Media, and Energy releases from Summer '17 to Summer '19 use the session cache for context rules. The cache allocation depends on the number of context dimensions used and varies from org to org.

To set the cache:

1. From Setup, in the **Quick Find** box, enter **`Platform`**.
2. Click **Platform Cache**.
   On the **Platform Cache Partition** page, if the CPQPartition appears, go to step 3.
   If the CPQ Partition does not appear, go to Create a Platform Cache Partition.



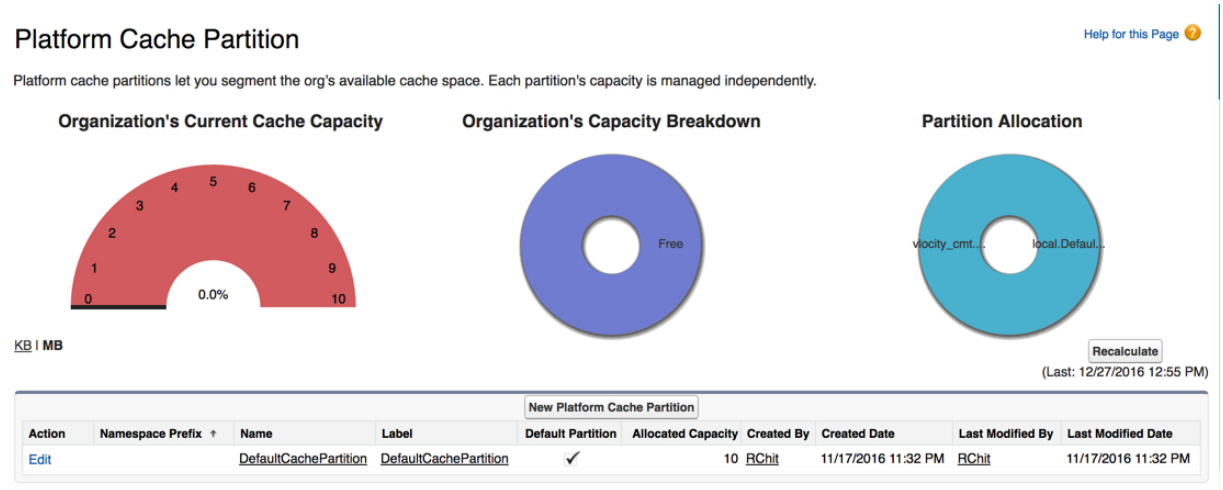
3. Next to CPQPartition, click **Edit**.
4. In the **Session Cache Allocation** list, in the **Organization** box enter a number between one and five. One is recommended.
5. In the **Org Cache Allocation** list, in the **Organization** box, enter a number between one and five. One is recommended.
6. Click **Save**.

Creating a Platform Cache Partition

Your managed package comes with a platform cache partition and you can create a new one if needed.

To create a platform cache partition:

1. From Setup, in the **Quick Find** box, enter **`Platform`**.
2. Click **Platform Cache**.



3. Click **New Platform Cache Partition**.
4. On the **New Platform Cache Partition** page, in the **Detail** section, enter the following information:
   - **Label** is **`CPQPartition`**.
   - **Name** is **`CPQPartition`**.
5. In the **Org Cache Allocation** list, in the **Organization** box, enter a number between one and the maximum cache allowed on your org.
   Org cache stores the query results for context mappings and dimensions, including changes made by CacheQueryStore. The Org Cache also contains rules information
6. In the **Session Cache Allocation** list, in the **Organization** box enter one of the following values:
   - Session Cache is no longer used for aggregate context mappings after CME Summer ’19.
   - For CME Fall ’19 and later, use a session cache value of 1.
7. Click **Save**.

Enable the CPQ Cache

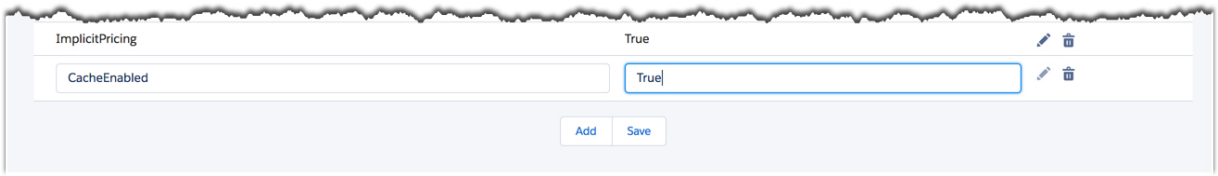
The CPQ cache comes enabled as part of the managed package. However, if you need to enable the CPQ cache, you first create the CPQ cache partition.

See Creating a Platform Cache Partition.

To enable the CPQ cache:

1. Go to the Vlocity CMT Administration tab and click **CPQ Configuration Setup**. To get to the Vlocity CMT Administration tab, click **App Launcher** or **All Tabs**, and then click **Vlocity CMT Administration**.
2. At the bottom of the CPQ Setup Configuration settings page, click **Add**.
   Two red fields appear.

3. In the box on the left, enter **`CacheEnabled`**.
4. In the box on the right, enter **`True`** or **`true`**.



5. Click **Save**.

Refreshing the Price Book Prior to Winter '18

NOTE
In Vlocity Communications, Media, and Energy Winter '19 and later, use the Refresh Platform Cache job on the Vlocity CMT Administration page to refresh price books. In Winter '18 and Fall '18, use the Refresh Pricebook job on the Vlocity CMT Administration page. For more information, see Administration Jobs Reference for Vlocity Communications, Media, and Energy.

You must refresh all price books that opportunities or orders use. While refreshing, Vlocity Communications, Media, and Energy processes the complete hierarchy, including the details of all product child items, and caches that information in the org level cache. Refreshing a price book also updates stored filterable attributes. If you do not refresh the price books after an upgrade, Vlocity Communications, Media, and Energy caches the hierarchy at run time when a product is accessed for the first time.

NOTE
You must refresh the price book in Salesforce Classic, not Lightning Experience.

Refreshing the price book stores filterable attributes and caches product bundles when the cache is enabled. You must refresh the price book in the following situations:

- When you add or modify an attribute or attribute value for any product in the price book
- When you add any new product to the price book, whether the product includes attributes or not
- When you add, modify, or delete the product child items for any bundle in a price book

NOTE
This is applicable only if you are using the platform cache. For more information, see Configure CPQ Platform Cache.

Refreshing the price book runs the following batch jobs:

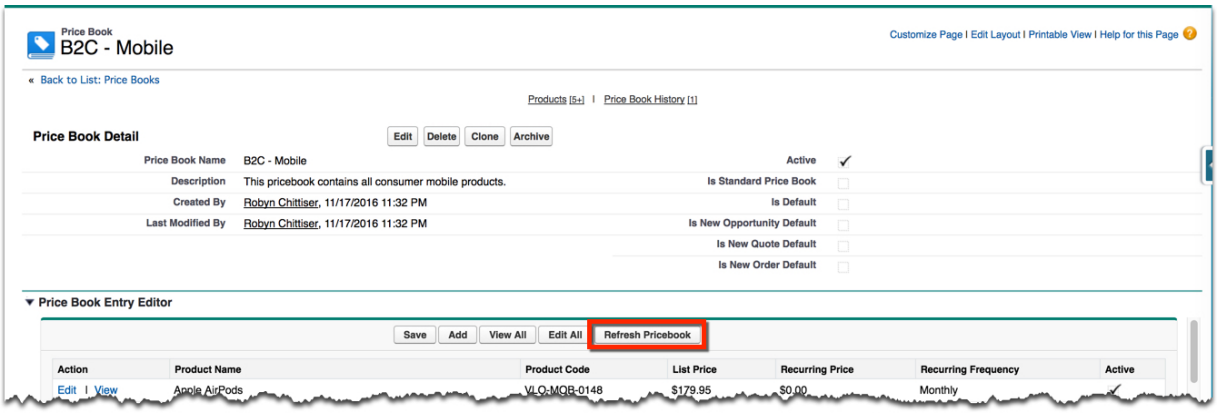
- The **ProductAttributesBatchProcessor** batch job copies the product attributes from the JSONAttribute field to the *sObject* Cached Filterable Attribute, enabling queries such as the filter in the Product List in Vlocity Cart.

- The **ProductHierarchyBatchProcessor** batch job copies the product hierarchies created in the **ResolveProductHierarchyBatchJob** to the CPQ partition platform org cache, leveraging the Salesforce platform cache for optimized product hierarchy operations and enables support for product hierarchies.

Do not refresh the price book while orders are being created. Invalid orders and runtime errors may result.

To refresh the price book:

1. Go to the Price Book tab.
2. Click the price book to refresh.
3. Click **Refresh Pricebook**.



> **NOTE**
> If the **Refresh Pricebook** button is not visible, you must expose it. For more information, see Expose the Refresh Pricebook Button

4. Repeat for each price book in your org.

Refreshing the Price Book in Winter '18 and Later

To refresh your price book in the Winter '19 release and later, run the Refresh Platform Cache job from the Vlocity CMT Administration page.

In all prior releases, use the Refresh Pricebook job on the Vlocity CMT Administration page.

For more information about admin jobs, see the Administration Jobs Reference for Vlocity Communications, Media, and Energy.

Configuring Power Update

Use Power Update to edit multiple line items at one time. When products are added to the cart, line items are created. When a bundle is added to a cart, the parent and child products are independent line items.

Starting from a line item or its fields, Vlocity can look up any other fields using the lookup relationship. For example, from a line item, you can look up a price book entry. From the price book entry, you can look up the product table and the product name field.

You can access Power Update settings from the Vlocity CMT Administration tab. Power Update settings are also located in the Field Settings custom setting.

To use Power Update, you must create specific field settings that include the following information:

- The Power Update panel in which the field appears
- The name of the field to edit, to use as a filter, or to display
- The name of the CPQ object, usually Opportunity, Order, or Quote

If there is an incorrect field setting, the Power Update user interface will not be available.

NOTE
Power Update is not compatible with Vlocity Cart.

To configure Power Update settings from the Vlocity CMT Administration tab:

1. On the Vlocity CMT Administration tab, click **Power Update**.
2. You can now create any of the following settings:
   - Display fields are the column names in the Power Update Records panel. To create display fields, see Create Power Update Display Fields.
   - Query fields are the picklist options in the Power Update Query panel. To create query fields, Create Power Update Query Fields.
   - Update fields are the picklist options in the Power Update Edit panel. To create update fields, Create Power Update Update Fields.

Create Power Update Display Fields

Display fields are the column names in the Power Update Records panel. In the images below, each of the column names in the Records panel corresponds to the fields in the Power Update Display Fields tab.

Figure 1. Power Update Records Panel

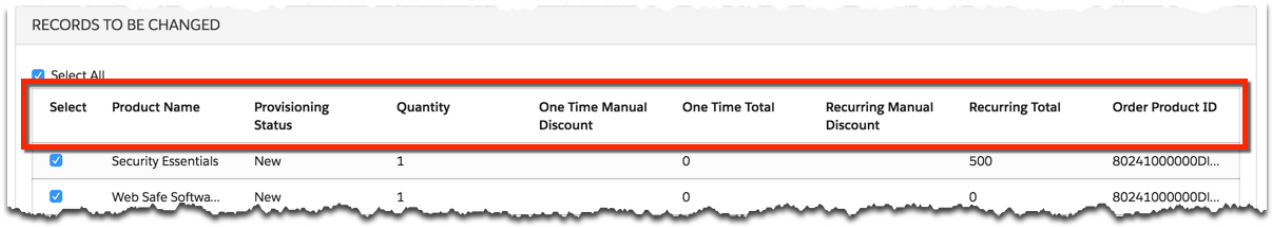
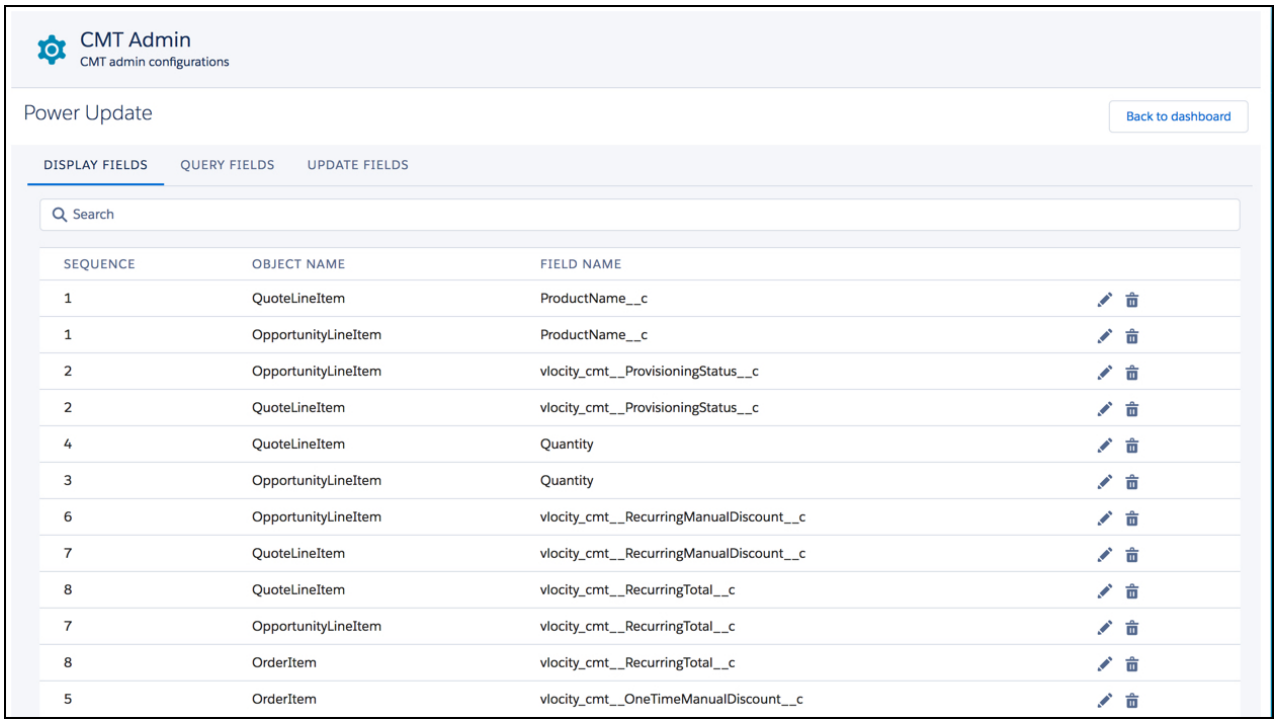


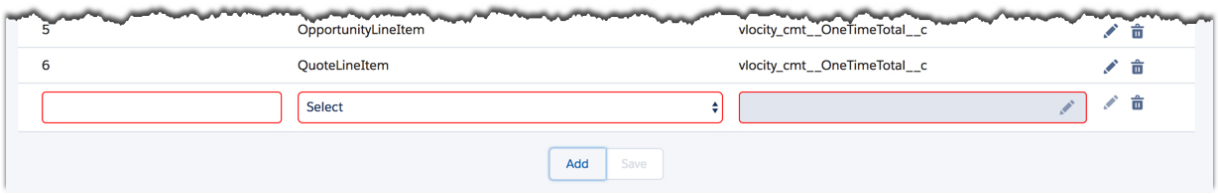
Figure 2. Vlocity CMT Administration Power Update Display Fields Tab



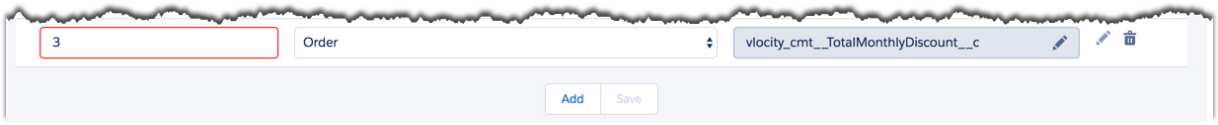
- Product Name corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem ProductName__c fields.
- Provisioning Status corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__ProvisioningStatus__c fields.
- Quantity corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem Quantity fields.
- One Time Manual Discount corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__OneTimeManualDiscount__c fields.
- One Time Total corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__OneTimeTotal__c fields.
- Recurring Manual Discount corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__RecurringManualDiscount__c fields.
- Recurring Total corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__RecurringTotal__c fields.

To create display fields:

1. On the Vlocity CMT Administration Power Update tab, click **Display Fields**.
2. At the bottom of the page, click **Add**.
   Two empty boxes and one picklist appear.



3. In the first red box, enter the sequence in which the column heading should appear.
4. From the picklist, select the object in which the field appears.
5. Click the **Action** icon .
   The **Field Selection** dialog box opens.
6. Select the field that contains the information to display.



7. Click **Save**.

Create Power Update Query Fields

Query fields are the picklist options in the Power Update Query panel. In the images below, each of the picklist values in the Query panel corresponds to the fields in the Power Update Query Fields tab.

The Query panel supports SOAP type string, Double, and Boolean values.

Figure 3. Power Update Query Panel

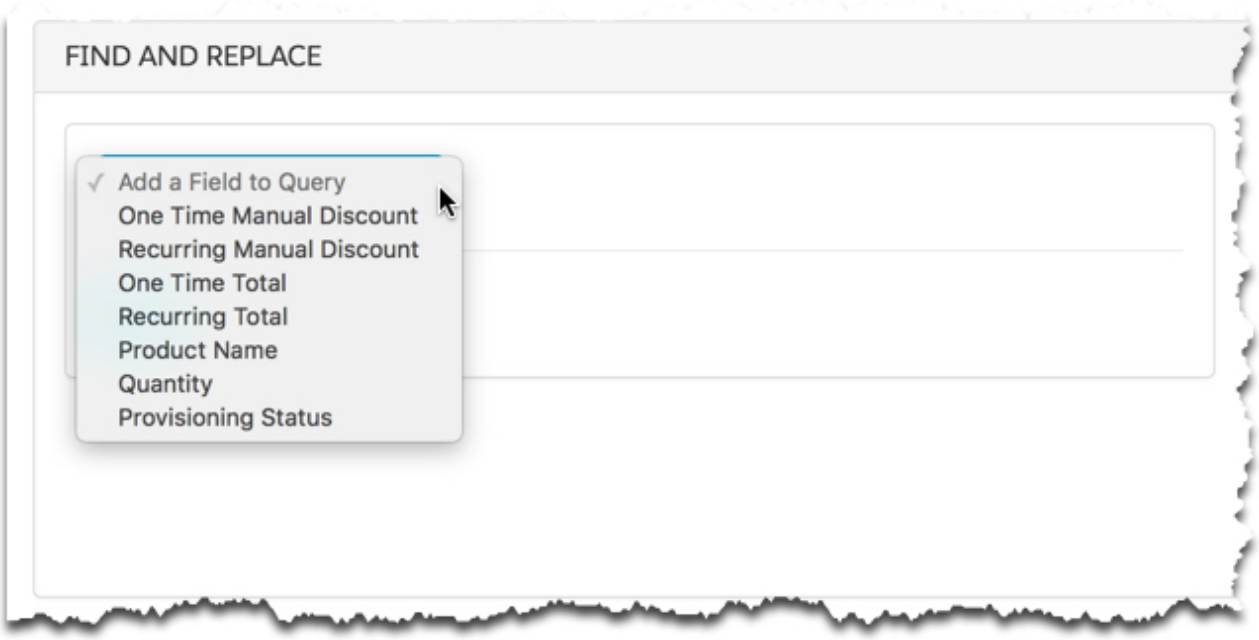
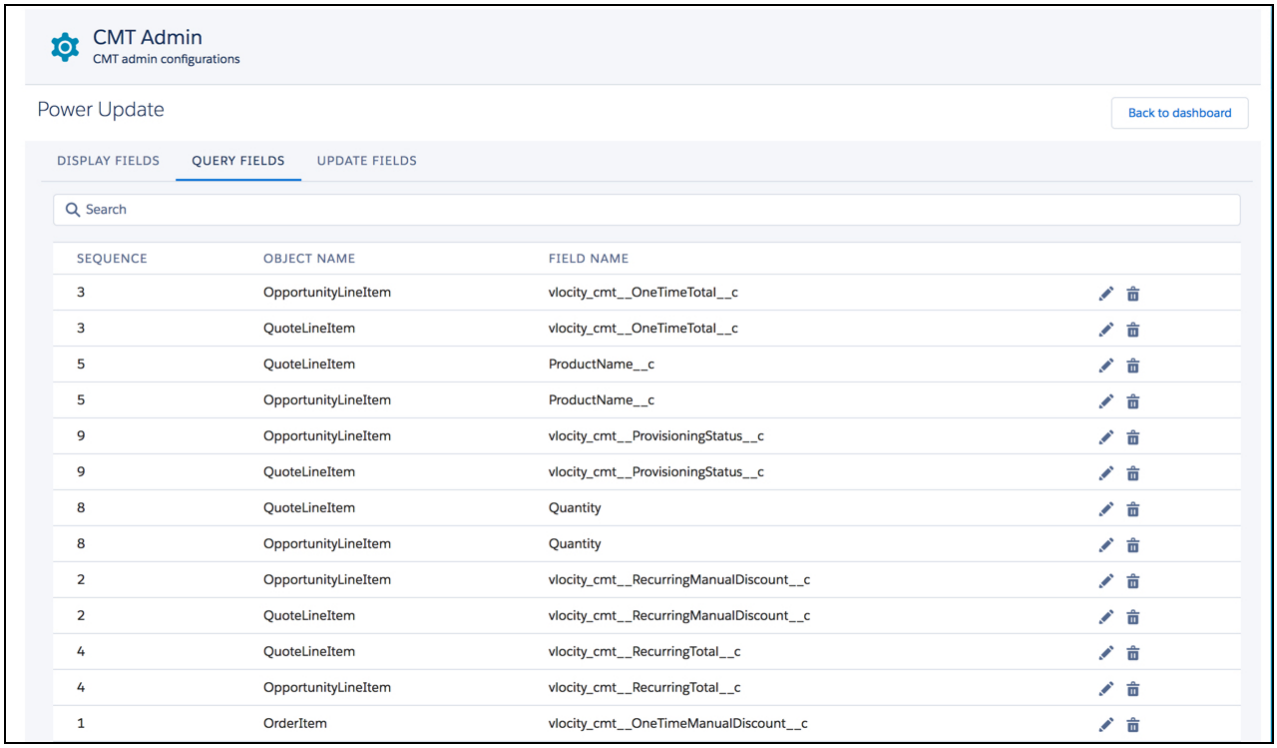


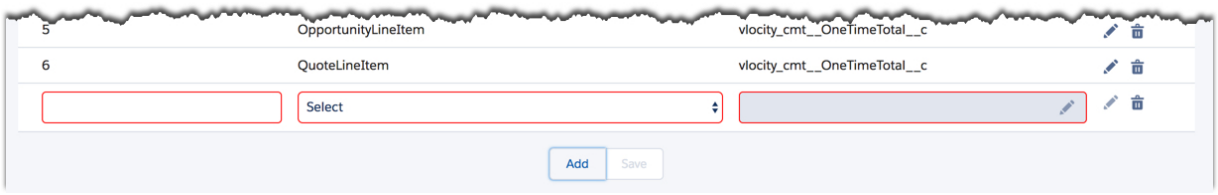
Figure 4. Vlocity CMT Administration Power Update Query Fields Tab



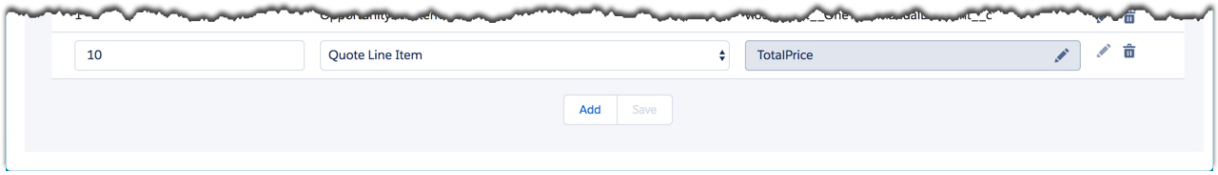
- One Time Manual Discount corresponds to the OpportunityLineItem, OrderItem, and QuoteLineItem vlocity_cmt__OneTimeManualDiscount__c fields.
- Recurring Manual Discount corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__RecurringManualDiscount__c fields.
- One Time Total corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__OneTimeTotal__c fields.
- Recurring Total corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__RecurringTotal__c fields.
- Product Name corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem ProductName__c fields.
- Quantity corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem Quantity fields.
- Provisioning Status corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__ProvisioningStatus__c fields.

To create query fields:

1. On the Vlocity CMT Administration Power Update tab, click **Query Fields**.
2. At the bottom of the page, click **Add**.
   Two empty boxes and one picklist appear.



3. In the first red box, enter the sequence in which the picklist item should appear.
4. From the picklist, select the object in which the field appears.
5. Click the **Action** icon .
   The **Field Selection** dialog box opens.
6. Select the field that contains the information to display.



7. Click **Save**.

Create Power Update Update Fields

Update fields are the picklist options in the Power Update Edit panel. In the images below, each of the picklist values in the Edit panel corresponds to the fields in the Power Update Update Fields tab.

The Edit panel supports string and double SOAP type values.

Figure 5. Power Update Edit Panel

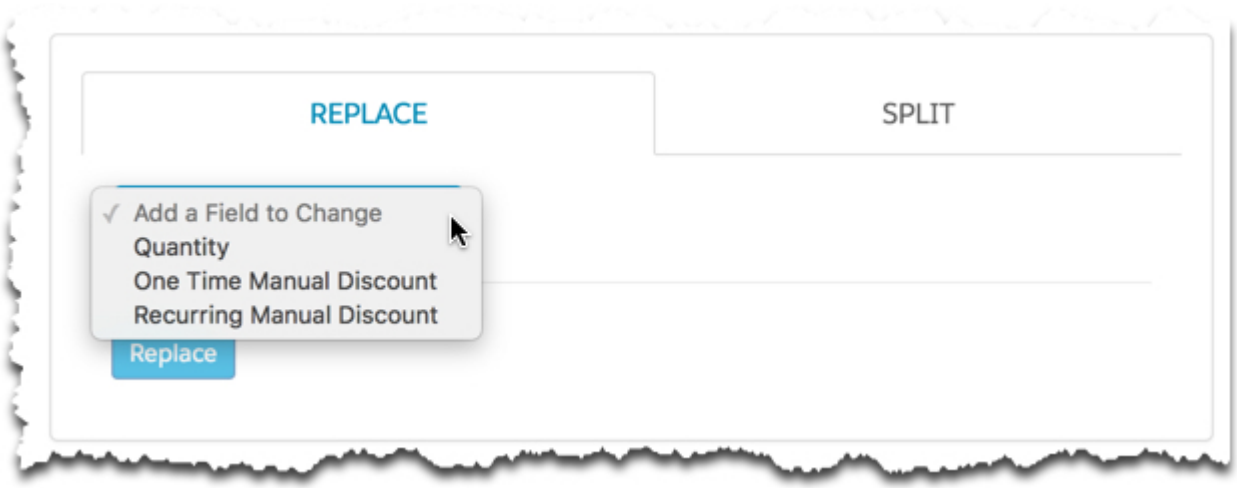
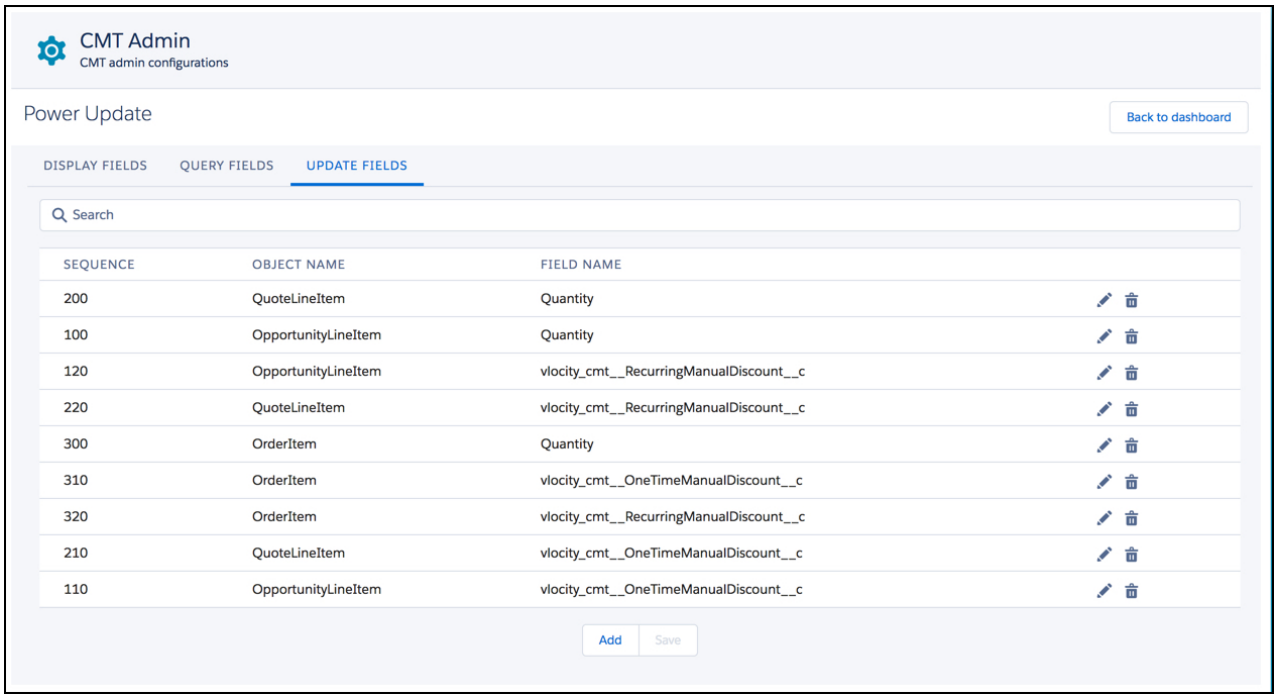


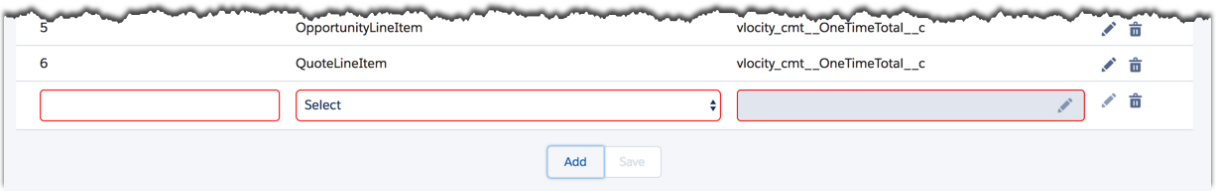
Figure 6. Vlocity CMT Administration Power Update



- Quantity corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem Quantity fields.
- One Time Manual Discount corresponds to the OpportunityLineItem, OrderItem, and QuoteLineItem vlocity_cmt__OneTimeManualDiscount__c fields.
- Recurring Manual Discount corresponds to the OpportunityLineItem, QuoteLineItem, and OrderItem vlocity_cmt__RecurringManualDiscount__c fields.

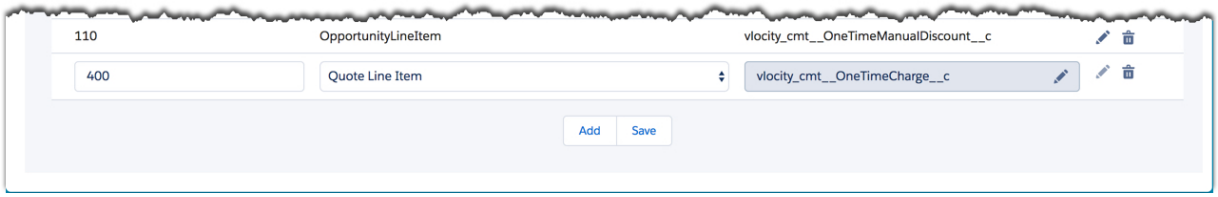
To create edit fields:

1. On the Vlocity CMT Administration Power Update tab, click **Update Fields**.
2. At the bottom of the page, click **Add**.
   Two empty boxes and one picklist appear.



3. In the first red box, enter the sequence in which the picklist item should appear.
4. From the picklist, select the object in which the field appears.
5. Click the **Action** icon .
   The **Field Selection** dialog box opens.

6. Select the field that contains the information to display.

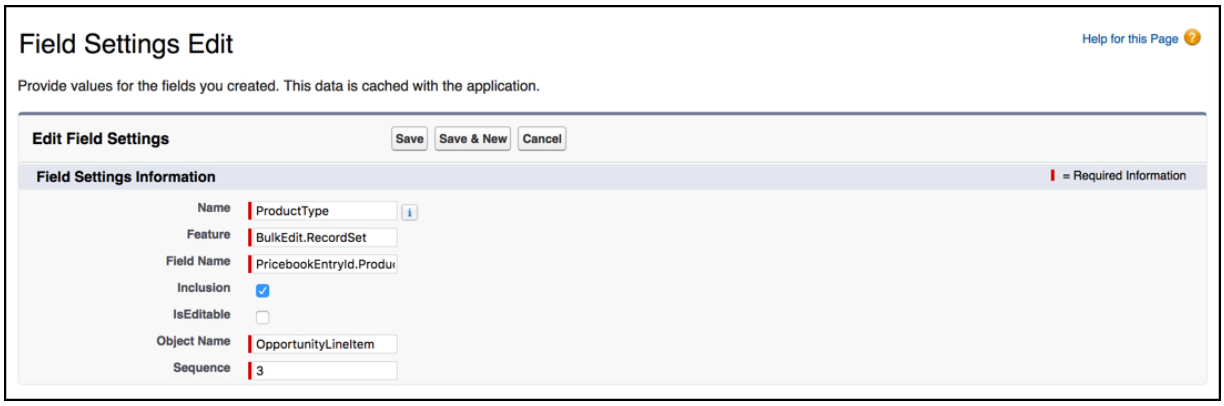


7. Click **Save**.

Configure Power Update Settings

To configure Power Update settings from the Field Settings custom setting:

1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. Click **Custom Settings**.
3. Next to Field Settings, click **Manage**.
4. Next to any of the setting names, click **Edit**.



5. Enter the following information:
   - **Name** is any string that names the setting object. Use this name to reference the custom setting in Apex.
   - **Feature** is one of three options:
     - **`BulkEdit.RecordSet`** creates a column name in the Records panel in Power Update.
     - **`BulkEdit.QuerySet`** creates an option in the Query panel picklist in Power Update.
     - **`BulkEdit.EditSet`** creates an option in the Edit panel in Power Update.
   - **Field Name** is the name of the field to update. When looking up from one table to another, you must use the format *`object.fieldname`*, for example, **PricebookEntryId.product2id.vlocity_cmt__Type__c**.
     In this example, Power Update starts in the line item table and fetches the PricebookEntryId value in this table. From the PricebookEntry table, using this ID, Power Update fetches the product2id value. Using the product2id, Power Update locates the product in the product2 table. After the product is identified, Power Update fetches the product type that uses the custom field vlocity_cmt__Type__c.
   - **Is Editable** is not used in Power Update settings.
   - **Inclusion** is not used in Power Update settings.

- **Object Name** is the name of the line item object to update. It is usually one of the following objects:
  - **`OpportunityLineItem`**
  - **`OrderLineItem`**
  - **`QuoteLineItem`**
- **Sequence** specifies the field sequence display.

The following is a list of some commonly used field settings:

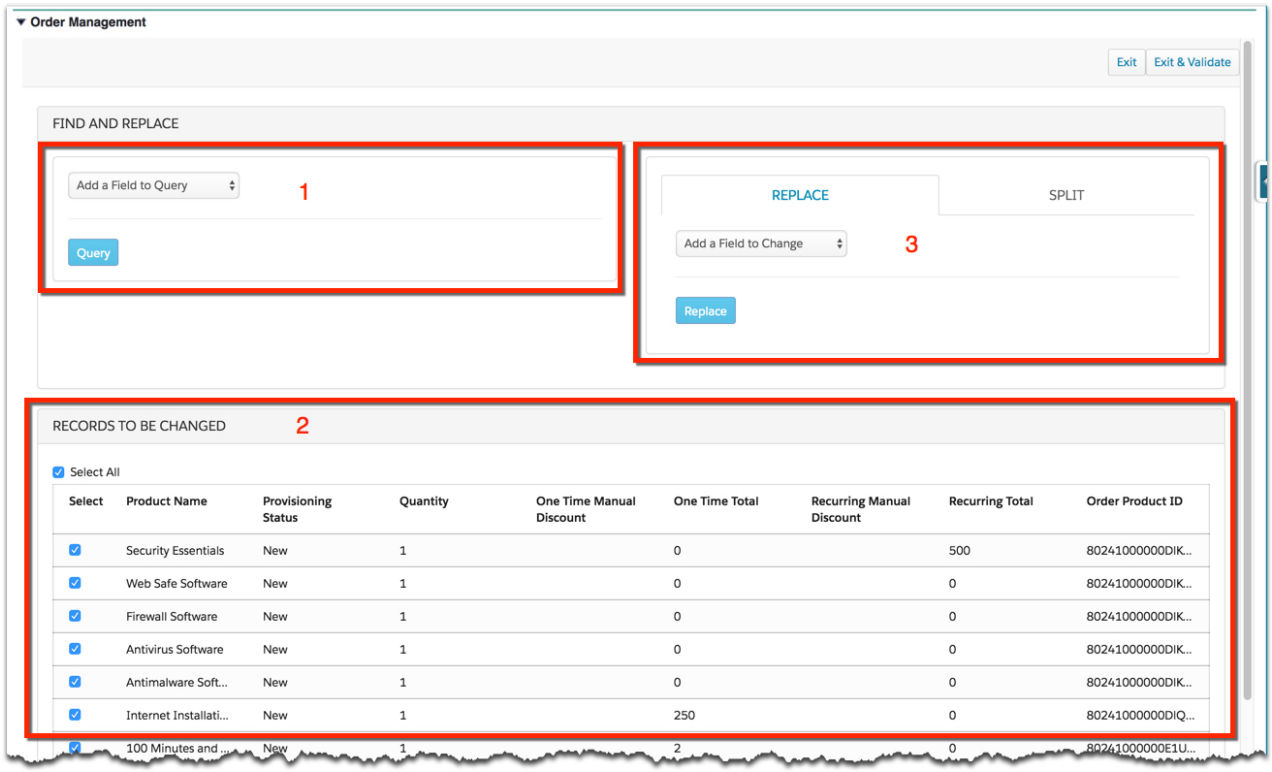
- PricebookEntryId.Product2id.Name
- Quantity
- PricebookEntryId.Product2id.Family
- PricebookEntryId.Product2id.vlocity_cmt__Type__c
- PricebookEntryId.Product2id.vlocity_cmt__SubType__c
- PricebookEntryId.Product2id.ProductCode
- PricebookEntryId.UseStandardPrice
- PricebookEntryId.UnitPrice
- PricebookEntryId.vlocity_cmt__RecurringPrice__c
- vlocity_cmt__OneTimeTotal__c
- vlocity_cmt__OneTimeManualDiscount__c
- vlocity_cmt__RecurringManualDiscount__c
- vlocity_cmt__ServiceAccountId__c
- vlocity_cmt__BillingAccountId__c
- PricebookEntryId.IsActive
- PricebookEntryId.Product2id.CanUseQuantitySchedule
- PricebookEntryId.Product2id.QuantityScheduleType
- PricebookEntryId.Product2id.QuantityInstallmentPeriod
- PricebookEntryId.Product2id.NumberOfQuantityInstallments
- PricebookEntryId.Product2id.CanUseRevenueSchedule
- PricebookEntryId.Product2id.RevenueScheduleType
- PricebookEntryId.Product2id.RevenueInstallmentPeriod
- PricebookEntryId.Product2id.NumberOfRevenueInstallments
- LastModifiedDate
- vlocity_cmt__LineNumber__c

Use Power Update

In the review cart, click the **Power Update** button to open the Power Update Visualforce page, which has three panels:

1. The Query panel filters the list of line items in the Records panel. You can query line items using multiple fields defined in the Field Settings custom setting. Choose a field setting, select a qualifying operator, and enter a value. Click the **Query** button to filter the line items in the Records panel.
2. The Records panel displays all of the line items in the cart. By default, all line items are selected. You can select any line items to modify. Power Update edits only the selected line items.

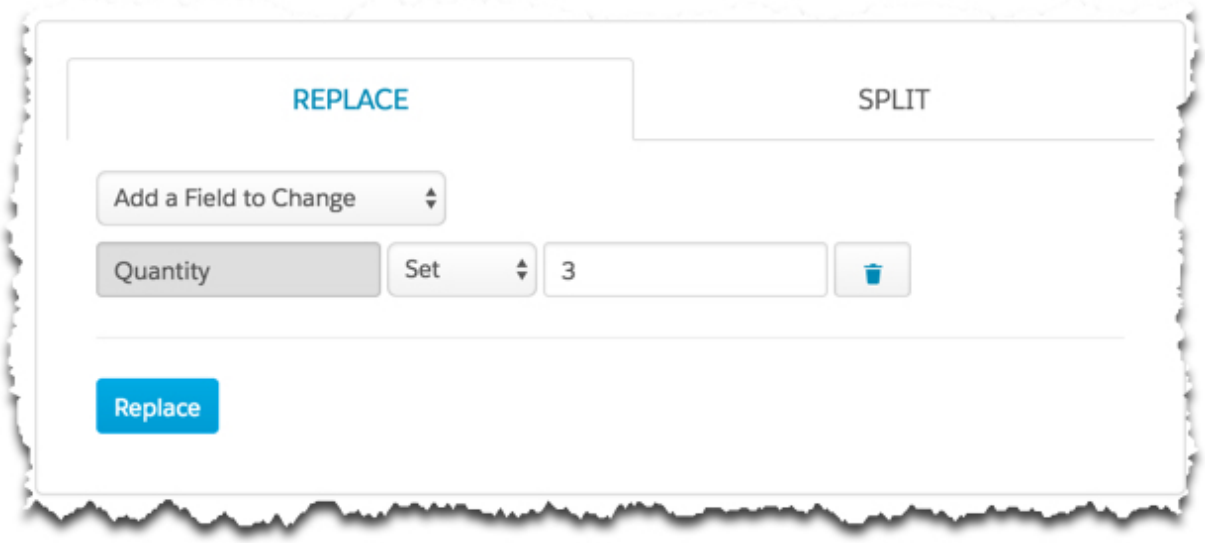
3. The Edit panel enables you to change the line items. Power Update interfaces directly with the database using *DML* requests. Other than standard data type checks and Salesforce behavior, the system does not validate the line items. After you click **Replace**, you cannot undo the operation. You can choose multiple fields to edit. Only selected line items are changed. Typical actions include set, increment, decrement, percent, multiply, and divide. After a successful operation, the records in the Records panel refresh. However, a refresh does not occur if an Apex job handles the operation.



To use Power Update:

1. Go to the review cart for an opportunity, order, or quote.
2. Click **Power Update**.
3. In the Query panel, from the **Add a Field to Query** picklist, select a field for which to search. The field name you selected appears in a line below the picklist.

4. Select an operator.
5. Enter a value for which to search.
6. To enter more search criteria, repeat steps 2 through 4.
7. Click **Query**.
8. In the Records panel, select the line items to change.
9. In the Edit panel, on the Replace tab, from the **Add a Field to Change** picklist, select the field to change.
   The field name you selected appears in a line below the picklist.



10. Select an operator.
11. Enter the value to replace the values in the selected line items.
12. Click **Replace**.
13. Click one of the following buttons:

- Click the **Exit** button to close the Power Update page. If any values were changed, they remain changed, but Power Update does not recalculate prices.
- Click the **Exit & Validate** button to close the power Update page and update the prices based on the changes.

NOTE
When an Apex job performs updates, Power Update automatically recalculates pricing.

Use Apex Jobs in Power Update

The DefaultPowerUpdateJobThreshold custom setting controls when an Apex job handles a Power Update operation. By default, if there are 200 or more line items, Power Update uses an Apex job. You can change the threshold for when to use an Apex job in Power Update.

To change the DefaultPowerUpdateJobThreshold custom setting:

1. From Setup, in the **Quick Find** box, enter **Custom Settings**.
2. Click **Custom Settings**.
3. Next to CPQ Configuration Setup, click **Manage**.
4. Next to DefaultPowerUpdateJobThreshold, click **Edit**.
5. In **Setup Value**, enter a new threshold.
   If the number of selected line items exceeds this threshold, an Apex job handles the Power Update operation.

   NOTE
   If you delete the value, there is no threshold; that is, an Apex job will never handle a Power Update operation.
6. Click **Save**.

When an Apex job handles a Power Update operation, Power Update notifies the user with the Apex job ID. The Records panel is not automatically refreshed when an Apex job runs. Power Update runs the job in multiple batches with a batch size of the threshold value.

To check the status of an Apex job, from Setup, in the **Quick Find** box, enter **`Apex jobs`**.

Split Line Items

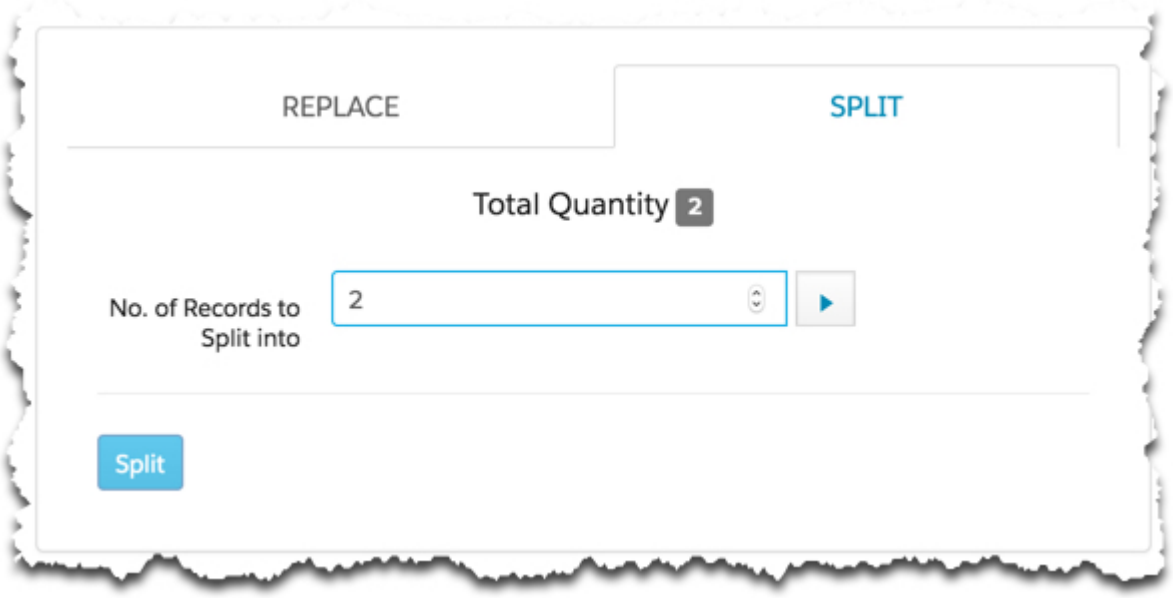
Power Update also includes the ability to split one line item with a quantity greater than one into multiple line items. The StrictSplitModeEnabled custom setting controls how items can be split.

To split a line item into multiple line items:

1. Go to the review cart for an opportunity, order, or quote.
2. In the Records panel, select the line item to split.

> **NOTE**
> The line item quantity must be greater than one.

3. In the Edit panel, click the Split tab.



4. Enter the number of records to split the line item into.
5. Click **Split**.
6. Click one of the following buttons:
   - Click the **Exit** button to close the Power Update page. If any values were changed, they remain changed, but Power Update does not recalculate prices.
   - Click the **Exit & Validate** button to close the power Update page and update the prices based on the changes.

Configure Line Item Splitting

You can split line items in Power Update. The StrictSplitModeEnabled setting controls how items can be split.

To configure how items are split:

1. On the Vlocity CMT Administration tab, click **CPQ Configuration Setup**.
2. In the StrictSplitModeEnabled row, click the **Action** icon .
3. In the **Value** box, enter one of the following options:
   - If the value is true, Power Update checks to ensure that the total quantity of all of the split line items add up to the original item quantity.

- If the value is false, the total quality of the split line items need not match the original quantity.

  For example, if you originally had two iPhones as one line item, if the StrictSplitModeEnabled setting is true, you can split the iPhones into two line items, no more, no less. However, if the StrictSplitModeEnabled setting is false, you can split the iPhones into as many line items as you wish.

4. Click **Save**.

You can also configure how items are split using the StrictSplitModeEnabled custom setting.

1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. Click **Custom Settings**.
3. Next to CPQ Configuration Setup, click **Manage**.
4. Next to StrictSplitModeEnabled, click **Edit**.
5. In the **Setup Value** field, enter false or true.

   If the value is true, Power Update checks to ensure that the total quantity of all of the split line items add up to the original item quantity.

   If the value is false, the total quality of the split line items need not match the original quantity.

   For example, if you originally had two iPhones as one line item, if the StrictSplitModeEnabled setting is true, you can split the iPhones into two line items, no more, no less. However, if the StrictSplitModeEnabled setting is false, you can split the iPhones into as many line items as you wish.
6. Click **Save**.

Configuring Cart Item Refresh Levels

As you add items to the cart, the cart contents increase. When the cart page refreshes, the `getCartsItems` API is called. The call can fail due to the heap size or an exceeded time limit. To support a large number of items when returning responses from `getCartsItems` and `addToCart`, Vlocity supports a level-based approach to refreshing the cart. This approach creates a smaller response to optimize performance. Use the level-based approach when you find that response times are excessive.

When you add an item to the cart or when you refresh the cart, instead of returning the complete product bundles, the level-based approach sends only the root along with the first level children. For example:

- Root
  - Parent 1
  - Parent 2
  - Parent 3

When you click the expand icon, the next level of child product is returned through a subsequent API call.

Required Settings

The following CPQ configuration settings control the level-based approach to cart item refreshing.

- LevelBasedApproach specifies whether to use the level-based approach to cart refreshing. If set to true, the value of the custom setting should be displayed when you create an order, add a bundle to the cart, and expand all the child items in the bundle.

- LevelBasedPageSize specifies the number of items that are returned on one page.
- LevelBasedPagination specifies whether to use the level-based pagination settings.

To use the level-based approach to refresh the cart, ensure that:

- Each custom field has a corresponding custom setting → Field Setting.
- The CacheEnabled setting is set to true.
- The DeltaPrice and DeltaValidate custom settings are added to your org.
- The DefaultHierarchy custom setting is greater than zero.
- You have specified any required parameters in the `getCartsItems` API. See Get Cart Items.

NOTE
You can also write a custom implementation of the `LevelBasedInterface` interface to control the behavior of level-based cart refreshing.

For instructions on creating custom fields, see Configure the Review Cart Using Custom Settings.

To configure the level-based approach to cart item refreshing:

1. On the Vlocity CMT Administration tab, click **CPQ Configuration Setup**.
2. In the LevelBasedApproach row, click the **Action** icon .
3. In the Value column, enter **`True`**.
4. In the LevelBasedPagination row, click the **Action** icon.
5. In the Value column, enter **`True`**.
6. In the LevelBasedPageSize row, click the **Action** icon.
7. In the Value column, enter an integer. The default value is 10.
8. Click **Save**. Now you need to ensure that each Custom Field in your org has a companion Field Setting.
9. Go to Salesforce Setup → Develop → Custom Settings.
10. Next to Field Settings, click **Manage**. The Field Settings page opens. If the field setting you want to add is already listed, namely a field corresponding to the Custom Field of interest, you are finished with this procedure. If the relevant field setting is not listed, you need to add it. In this case, go to the next step.
11. Click **New** and then add the required information.
12. Ensure that **Inclusion** is selected.
13. Click **Save**.
14. For each field setting you need to add, repeat steps 10 to 13 .

You are finished configuring the level-based approach to cart item refreshing.

Running the EPCProductAttribJSONBatchJob

The `EPCProductAttribJSONBatchJob` regenerates the Product object JSONAttribute fields for products that were created and are managed in Product Console. Run the `EPCProductAttribJSONBatchJob` to sync the Product object JSONAttribute fields with the attributes that are assigned to the products.

Run this job when you add an attribute to an existing object type that is associated with products.

When running the `EPCProductAttribJSONBatchJob`, you enter a script to get a list of Product2 IDs, or `prod.Ids`.

To run the `EPCProductAttribJSONBatchJob`:

1. From the **User** menu, choose **Developer Console**.
2. From the **Debug** menu, choose **Open Execute Anonymous Window**.
3. Delete any code that appears in the **Enter Apex Code** dialog box.
4. Paste the following statement into the **Enter Apex Code** dialog box, replacing *`vlocity_cmt`* with your namespace:

```
List<Id> productIds = new List<Id>();
for (Product2 prod : [ Select Id from Product2 where
vlocity_cmt__ObjectTypeId__c != null ])
{
productIds.add(prod.Id);
}
Database.executeBatch(new
vlocity_cmt.EPCProductAttribJSONBatchJob(productIds), 1);
```

5. Click **Execute**.
6. Close Developer Console.

When you import data that contains product attributes, such as when creating a sandbox environment, you must correct the product attribute record IDs. If your org contains migrated product attribute overrides from promotions, you must also run EPCFixCompiledAttributeOverrideBatchJob to correct the attribute record IDs.

Analyzing CPQ Processing with Vlocity Tracking Service

As a performance engineer or system integrator working on customer projects, you can analyze the timing of CPQ processing to quickly drill into the problem areas where CPQ processing is slow. Using the tracking service greatly speeds-up performance analysis of CPQ processing.

For example, you can track the following data points for CPQ APIs:

```
{ "TrackingService": "CPQ", "TrackingEvent": "getCarts",
"SalesforceSessionToken": "11312ae20cc8957a3fa0cb8360a4e407", "UserId":
"0050Y000000Dp0zQAC", "Timestamp": "2016-12-14T12:18:15+00:00", "ElapsedTime":
5956, ...insert other CPQ-related info here... }
```

Where `"CPQ"` is the TrackingService.

To enable this functionality, Vlocity Tracking Service is implemented with Vlocity Cart Validation and tracks the following fields for each CPQ API call:

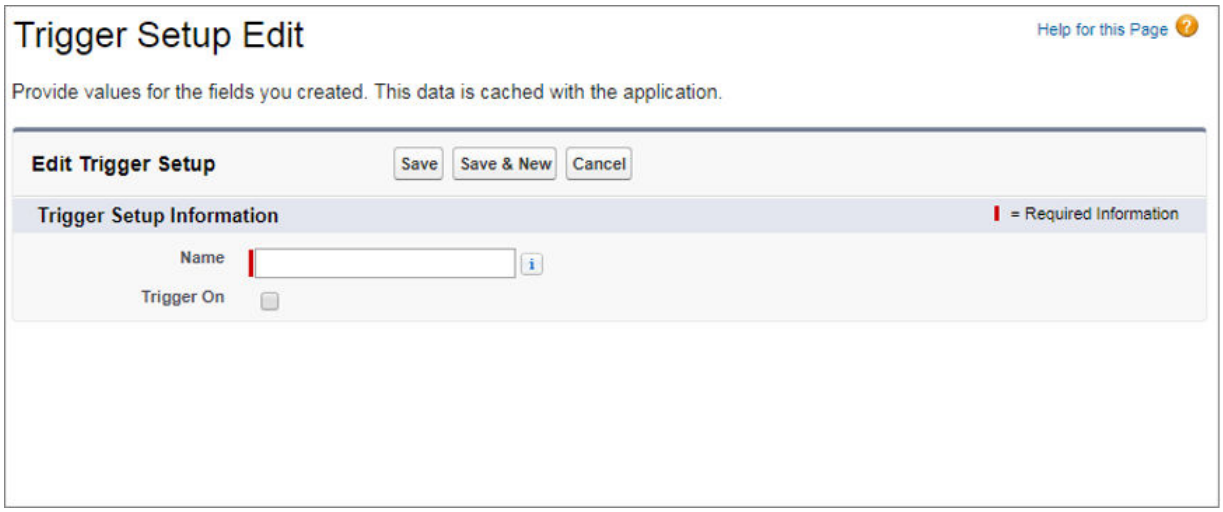
Table 1. Tracked Fields

Field	Value
TrackingService	CPQ
TrackingEvent	methodName – `getCarts`, `postCartsItems`
ClassName	`CPQAppHandler`
ElapsedTime	Time taken for this API call
CartId	Cart ID passed in the request, if the ID was passed in the input. Otherwise, the value is null.

For more information, see Vlocity Tracking Service.

To enable CPQ tracking, you add a trigger with an appropriate name. For example, to enable tracking for `getCarts` API, create a trigger, name it **`Track.CPQ.getCarts`**, and set it to **On**.

1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. From the search results, click **Custom Settings**. The Custom Settings page appears.
3. Next to Trigger Setup, click **Manage**. The Custom Setting page for the Trigger Setup appears.
4. Click **New**. The Trigger Setup Edit page appears.



5. Set the triggers:
   - To enable tracking for all CPQ APIs, specify the following:
     - **Name**: **`Track.CPQ`**
     - **Trigger On**: Select to set the trigger to On.
   - To enable tracking for only one CPQ API, specify the following:
     - **Name**: **`Track.CPQ.<methodName>`**
     - **Trigger On**: Click the checkbox to set the trigger to On.
6. Click **Save**.

The tracking data is stored in the Vlocity Tracking Entry object (VlocityTrackingEntry__c). For more details, see Vlocity Components Event Tracking.

You can have many triggers simultaneously that enable tracking for different APIs. You can also enable tracking at the CPQ level or at the individual method level.

Enabling Compatibility Using the Cart Customizable Attribute Level Setting

In Vlocity Communications, Media, and Energy Summer '17, the Vlocity Cart UI introduces an enhancement that changes how the Vlocity Cart determines which attributes are configurable in the cart. However, to enable customers to choose when to switch to the new behavior, the **Cart Customizable Attribute Level** custom setting is provided to enable compatibility with prior releases.

- For compatibility with Vlocity Communications, Media, and Energy Spring '17, set the **Cart Customizable Attribute Level** custom setting to **Attribute**.
- When you are ready to switch to the new, enhanced Vlocity Cart behavior, set the **Cart Customizable Attribute Level** custom setting to **Product Attribute**. You could also delete the custom setting.

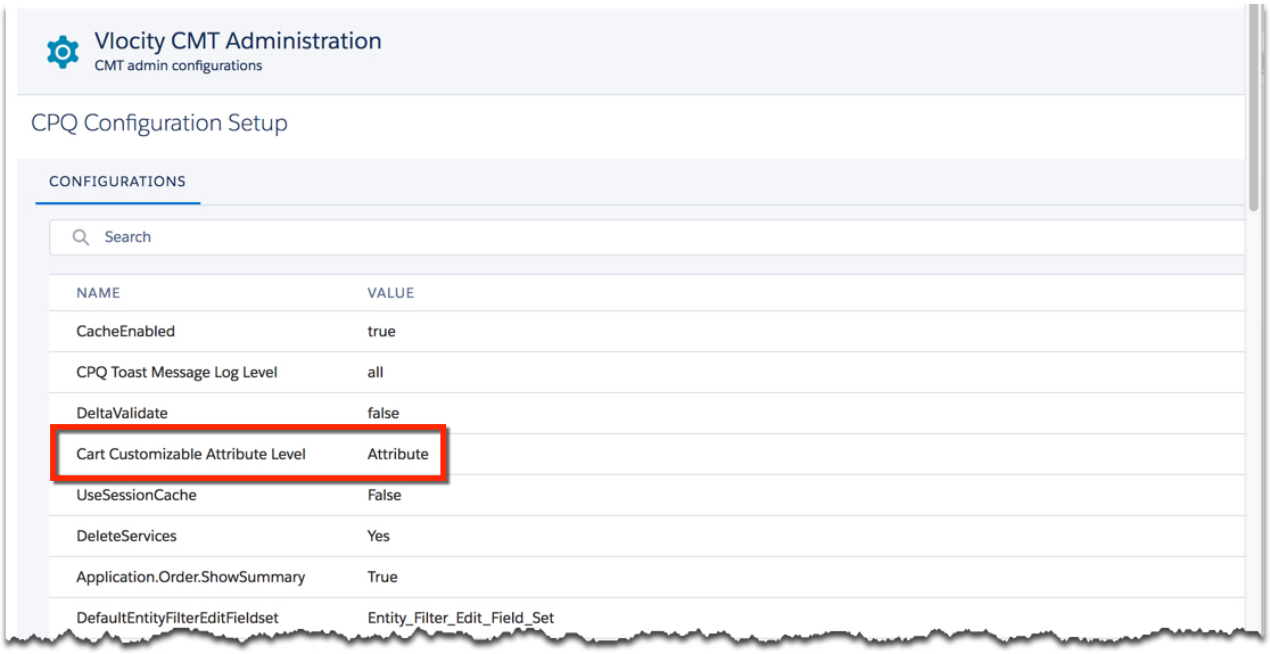
When installing the Vlocity Communications, Media, and Energy Summer '17 package for the first time, the **Cart Customizable Attribute Level** custom setting is added and set to **Product Attribute**.

NOTE

In future releases, the **Cart Customizable Attribute Level** setting may not be installed, because the absence of the setting enables the new behavior.

When upgrading the Vlocity package from Vlocity CMT Spring '17 to Vlocity Communications, Media, and Energy Summer '17, Vlocity automatically adds the **Cart Customizable Attribute Level** custom setting and sets it to Attribute. The Vlocity Cart behavior does not change. When upgrading, confirm that the custom setting has been added. On the Vlocity CMT Administration tab, click CPQ Configuration Setup. The **Cart Customizable Attribute Level** should be listed.

If it does not appear, you must add it. For more information, see Add CPQ Configuration Setup Settings. You can remove it later.

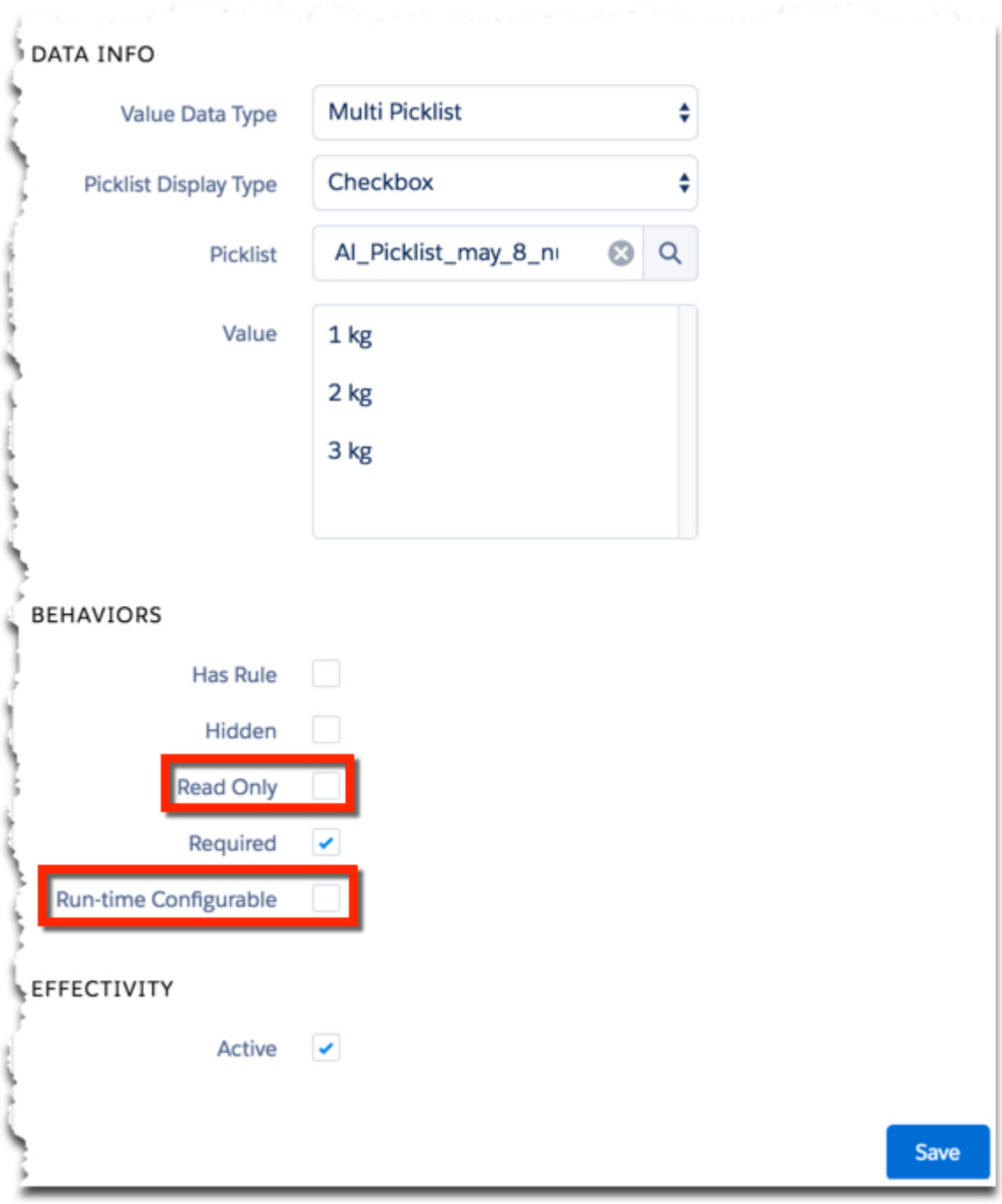


In Vlocity CMT Spring '17 (V15), the product attribute **Read-Only** behavior setting controls whether the attribute is configurable in quotes and orders in Vlocity Cart.

In Vlocity Communications, Media, and Energy Summer '17, both the **Read-Only** behavior setting and the **Run-time Configurable** behavior setting control whether the attribute is configurable. This enhancement provides better control over customizing product attributes.

The **Read-Only** and **Run-time Configurable** settings appear when configuring individual product attributes. For more information, see Product Attributes Overview.

Figure 7. Vlocity Product Console, Attribute Metadata, Behavior Settings



If the **Run-time Configurable** setting is false, the value is fixed at design time. Users cannot configure the attribute in quotes and orders regardless of how the Read Only setting is set.

If the **Run-time Configurable** setting is true, the value is not fixed. Whether users can configure the attribute value in quotes and orders depends on how the Read Only setting is set.

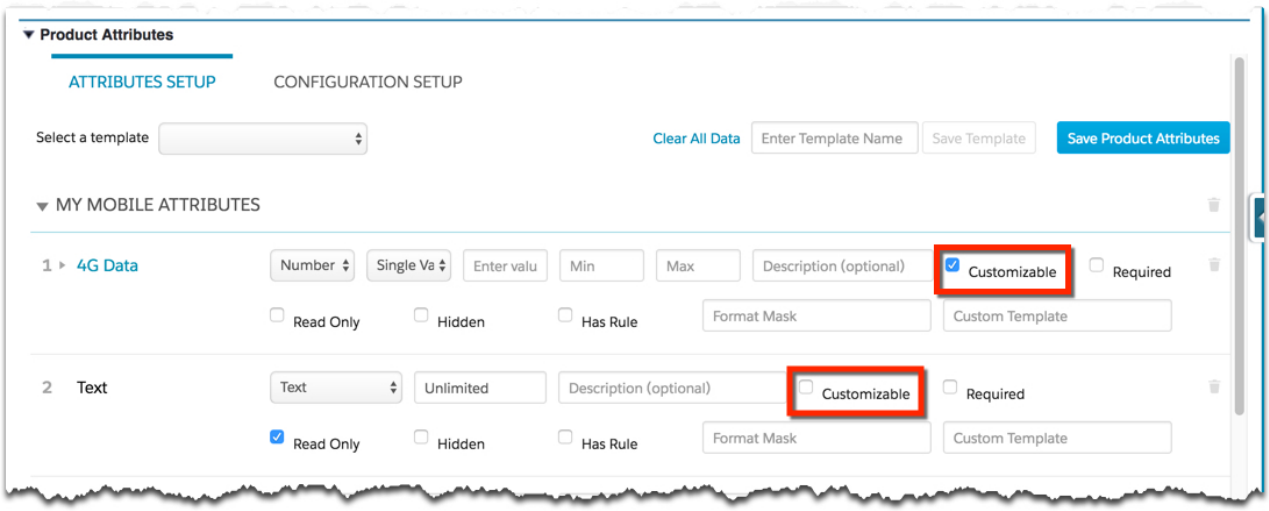
- If the **Read Only** setting is false, users can directly edit the attribute value during order capture.
- If the **Read Only** setting is true, users cannot directly edit the attribute value during order capture. Instead, the value might be set at order capture through an API call, calculation, or process, such as a process to assign an IMSI value to a new mobile plan.

Customers upgrading from Vlocity Spring '17 (V15) who are using Vlocity Cart will likely have many products with the **Read Only** setting false, and the **Run-time Configurable** setting false. If the **Cart Customizable Attribute Level** setting is not set to **Attribute**, these attributes become read-only for all components in Vlocity Cart.

By default, upgrading customers have the **Cart Customizable Attribute Level** setting set to **Attribute** for compatibility with Vlocity CMT Spring '17. Vlocity strongly recommends that customers enable the enhanced control over attribute customization by changing the **Run-time Configurable** setting to true for all product attributes outside of the production org. Then, set the **Cart Customizable Attribute Level** setting to **Product Attribute**. Test that all components are functioning as expected in Vlocity Cart. When the testing is successful, push the product attribute changes and the new **Cart Customizable Attribute Level** setting to production. At that time, you can remove the **Cart Customizable Attribute Level** setting.

The **Run-time Configurable** setting in the EPC Product Console is the same as the Customizable setting through the Salesforce Product record pages. On the Salesforce Product record pages, the Customizable setting was originally used only to enable the additional Picklist data type for a product attribute. The Customizable setting continues to behave that way in the Salesforce Product record pages, but also takes on the new Vlocity Communications, Media, and Energy Summer '17 functionality as well. For more information, see Product Attributes Overview.

Figure 8. Product Record Detail Page, Product Attributes, Customizable Setting



Creating a Custom Product Configuration Message

Product compatibility rules, also known as product configuration rules, determine if a product combination is valid. Vlocity includes default product configuration error messages in the Opportunity Manager, Order Manager, Quote Manager, and cart. However, you can create custom messages to provide more useful information to the user.

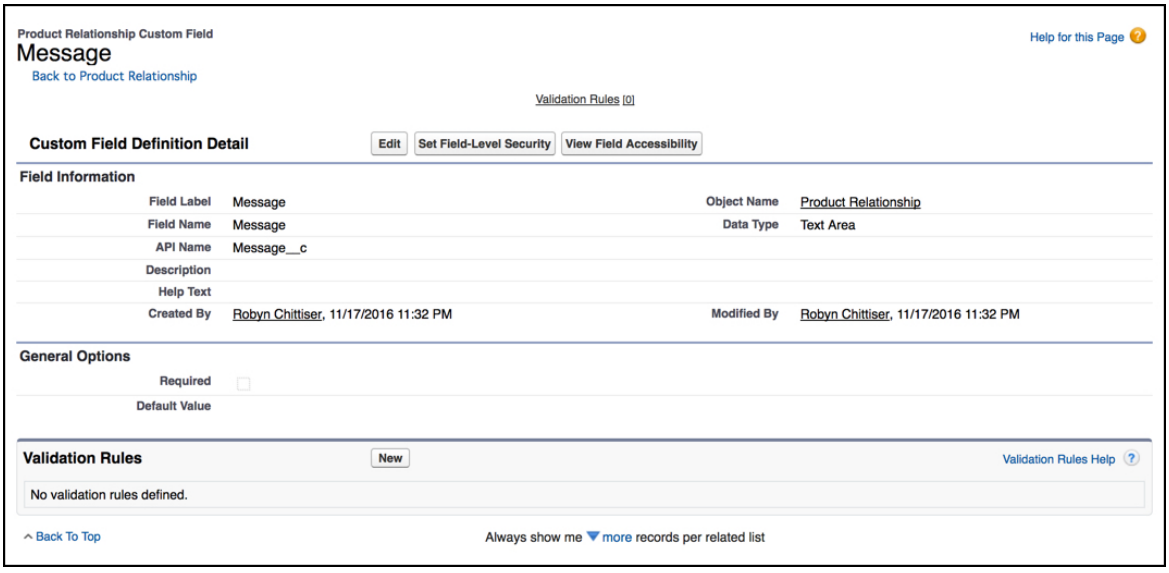
For example, consider these sample error messages:

- Product X requires Product Y
- Product X excludes Product Y
- Product X recommends Product Y

You can create a custom message for each product relationship record. First, you create a user custom message field to display instead of the standard message. Then, you set the CustomRuleMessageFieldName custom setting.

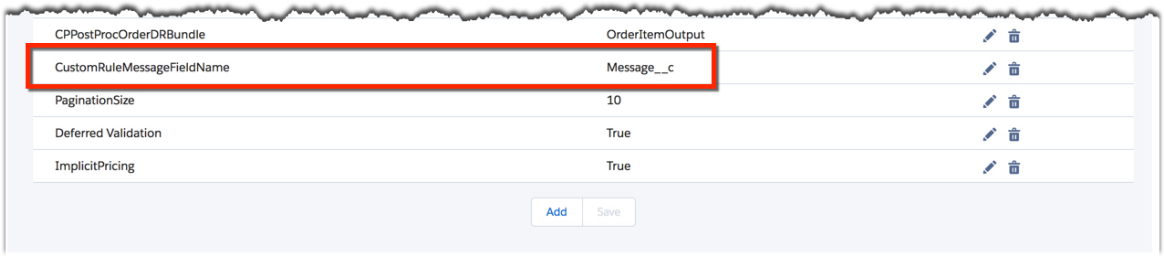
To create a custom product configuration message:

1. Create a user custom message field.
   a. From Setup, in the **Quick Find** box, enter **`Objects`**.
   b. Click **Objects**.
   c. Click **Product Relationship**.
   d. In the **Custom Fields & Relationships** section, click **New**.
   e. Create a new custom field with a **Data Type** of **Text Area**. Remember the API name for the field. For more information, see Create Custom Fields in the Salesforce Help.

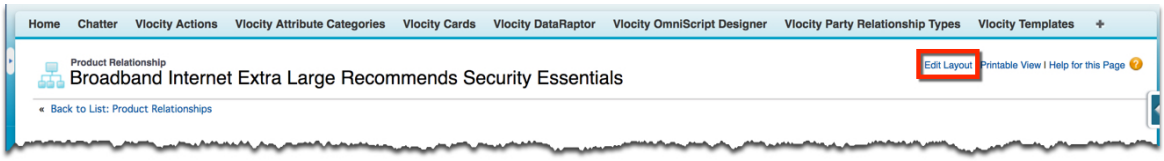


2. Configure the custom field.
   a. On the Vlocity CMT Administration tab, click **CPQ Configuration Setup**.
   b. Scroll to **CustomRuleMessageFieldName**.

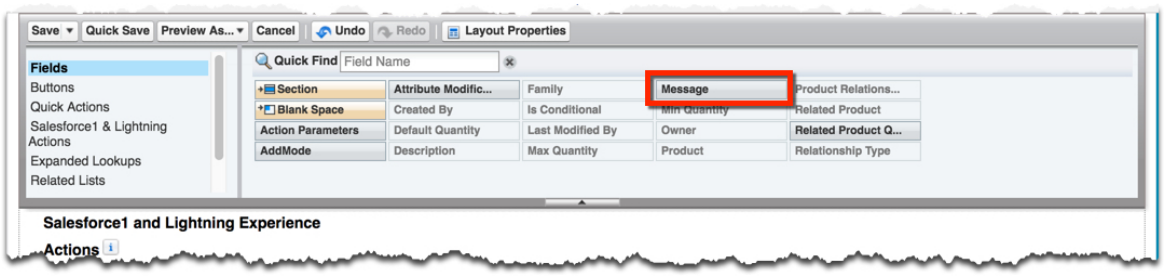
c. Click the **Action** button.
d. Enter the API name from the field you created in step 1, for example, **`Message__c`**.
e. Click **Save**.



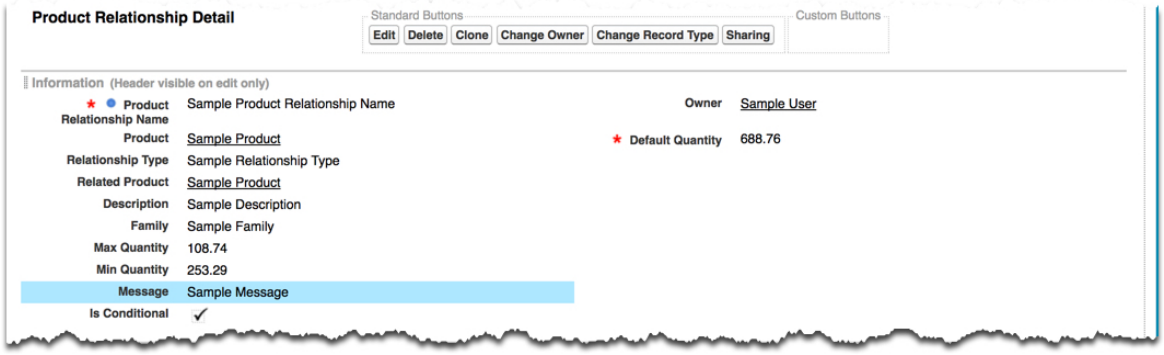
3. Expose the new field on the Product Relationships record detail page.
   a. On the Product Relationship tab, click a product relationship.
   b. Click **Edit Layout**.



   c. In the palette, select **Fields**.

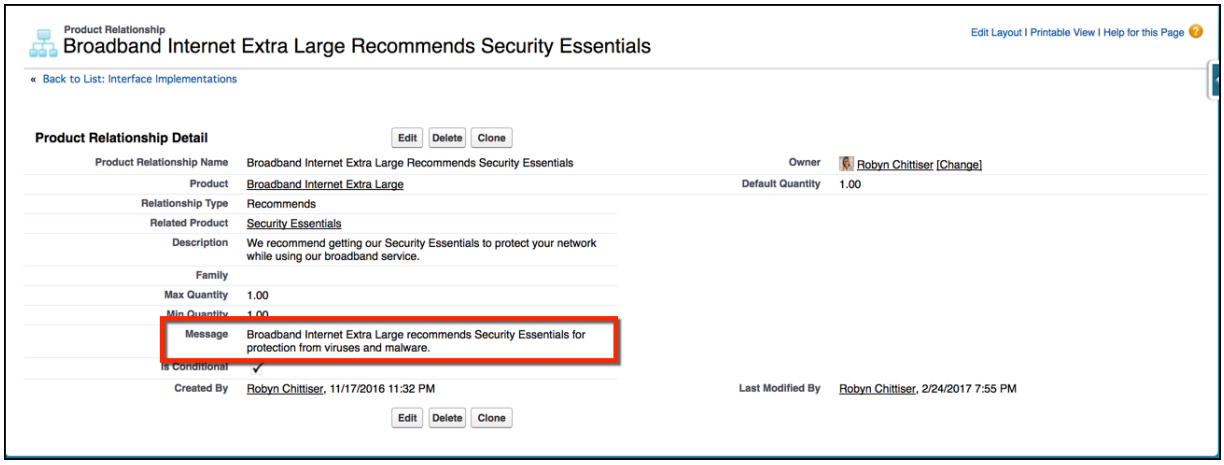


   d. Drag the field you created in step 1 to the **Product Relationships Detail Information** section.

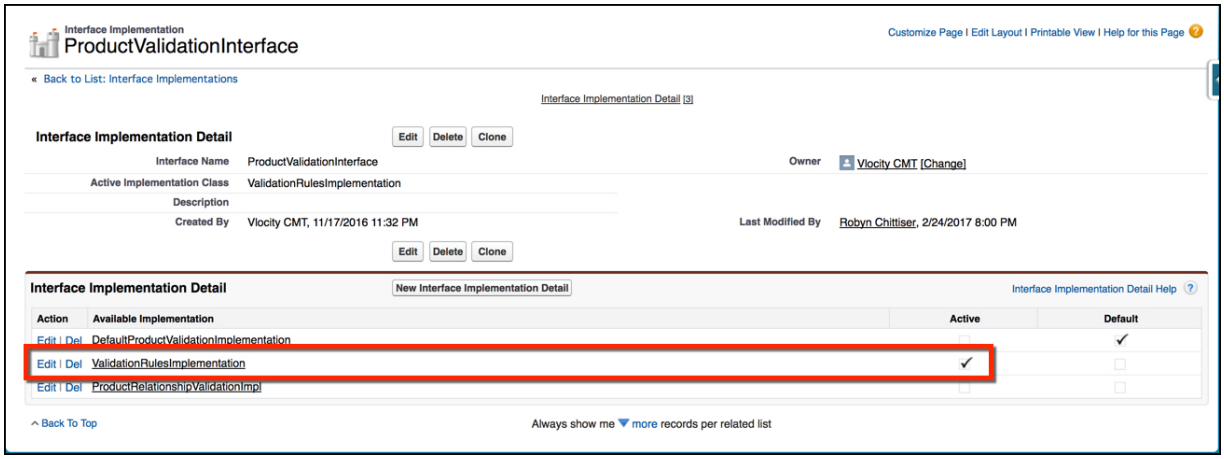


   e. Click **Save**.
4. Enter the custom message.
   a. On the Product Relationship tab, click a product relationship.

b. In the custom field, for example, **Message**, enter the custom message.



5. Ensure that the active implementation for the ProductValidationInterface is a validation rule implementation.
   a. On the Interface Implementations tab, from the **View** picklist, select **All**.
   b. Click **Go!**.
   c. Click **ProductValidationInterface**.
   d. Ensure that ValidationRulesImplementation or a custom implementation that invokes compatibility rules is active. If it is not, click the appropriate implementation and select **Active**.



Adding the ConfigureWithNamespace Button to Layouts

In Lightning mode, the ConfigureWithNamespace button has been added to the Order, Opportunity, and Quote layouts as a custom button by default. The Configure custom button no longer works. If upgrading from CME Winter ’19 to CME Summer ’19 or later, you can configure your own button to invoke CPQ using the Object Manager to add **ConfigureWithNamepsace** to the layout manually.

To add the **ConfigureWithNameSpace** button to a page layout:

1. In Lightning Experience, from Setup, in the **Quick Find** box, enter **`Object Manager`**.
2. Click **Object Manager**.
3. Click the object to which to add the button—**Opportunity**, **Order**, or **Quote**.
4. Click **Page Layouts**.
5. Click the page layout for your object record detail page, for example **Order (Vlocity Telecommunication Services) Layout**.
6. In the palette, click **Buttons**.
7. Drag the **ConfigureWithNameSpace** button from the palette to the `Object` **Detail**, **Custom Buttons** section.
8. Click **Save**.

Configuring Values for Status and Order Status Fields

When you create an order and submit it to Order Management for processing, the status of the order is changed to "In Progress". When you create a subsequent order on the same asset while the first order is still in progress, the changes made in the first order are stacked into the second order. For example, if the subsequent order is scheduled and submitted after the original order is completed, it can include the changes from the order in progress. This is achieved through a flexible FDO check so that additional Order Status values are considered when looking for FDOs.

In Asset to Order FDO MACD flow, Order's Status and OrderStatus__c fields query for a previously created MACD order. The query uses the following values:

Field	Values in Query
Status	Draft
OrderStatus__c	Ready To Submit,Queued

The FDOStatuses setting lets you configure the possible list of values for Status and OrderStatus__c fields.

Format

The value for the status FDOStatuses is a semicolon-separated list of fields and their corresponding values:

```
<FieldAndValue>;<FieldAndValue>
```

To specify only one FieldAndValue, you need not use a semi-colon. Each FieldAndValue is an expression in this format:

```
<Field>=<Comma-separated-values>
```

Examples

The following table lists FDOStatus values for various use cases.

Example	Value for FDOStatuses
To add Active to the list of values for an order's Status field and leave the values for OrderStatus__c unchanged	Status=Draft,Active

Example	Value for FDOStatuses
To add In Progress to the list of values for Order's OrderStatus__c field and leave the values for Status unchanged	OrderStatus__c=Ready to Submit,Queued,In Progress. Specify all values, including the default value, because the setting completely overrides the default value. To add a new OrderStatus__c value, specify the default values in addition to the new value.
To add Active to Status and add In Progress to the OrderStatus__c field	Status=Draft,Active;OrderStatus__c=Ready to Submit,Queued,In Progress
To use default behavior	Do not configure the property.

In Vlocity CME, the CPQ Order Status field (API Name: vlocity_cmt__OrderStatus__c) includes values such as Ready To Submit, In Progress, Activated, Cancel Requested, Cancelled, Completed and so on. See Order Statuses in CPQ.

The Vlocity OM Fulfilment Status field (API Name: vlocity_cmt__FulfilmentStatus__c) includes values such as Draft, In Progress, Activated, and so on.

IMPORTANT

DO NOT customize the Order Status or Fulfilment Status fields to extend the picklist values for your custom project values. Instead, create a custom field with the required milestones as picklist values. See Creating a Custom State Field.

Enable Usage Pricing

Before you start using the Multi-Site Quote and Order Capture feature, you must enable usage pricing.

To enable usage pricing:

1. Log in to Salesforce and go to the App Launcher.
2. Enter Vlocity CMT Administration Console in the quick find search.
   The Vlocity CMT Administration Console is displayed.
3. From the Custom Settings section, select Enable Features.
4. Navigate to the **Usage Pricing** feature and toggle the switch to enable it.
   You will see a confirmation message.
5. Click Enable.
   You will see a confirmation message indicating that the feature is enabled.
6. Click OK.

After you enable the usage pricing feature, the following options are available in Vlocity Cart.

NOTE
If you do not enable the Cost and Margin feature, Usage Pricing is only available in the drop-down menu of Vlocity Cart.

Table 2. Usage Pricing Fields Available in Vlocity Cart

Vlocity Cart Tab	Available Columns with Usage Pricing Enabled	Available Columns with Usage Pricing and Cost and Margin Enabled
When you enable the usage pricing feature and the cost and margin feature, you will see Usage Pricing + Cost and Margin in the drop-down menu in Vlocity Cart.	Usage Pricing	Usage Pricing + Cost and Margin
New columns for each line item in the Cart are available when you enable Usage Pricing. These are calculated based on the usage pricing variables you defined.	• Usage Unit of Measure • Usage Price • Usage Price Total	• Usage Margin • Usage Price • Usage Price Total • Usage Unit of Measure
New columns in the Total Bar are available when you enable Usage Pricing. These are calculated based on the usage pricing variables you defined.	• Group Usage Total • Group Recurring Total • Group One Time Total	• Group Usage Total • Usage Margin Total • Group Recurring Total • Recurring Margin Total • Group One Time Total • One Time Margin Total • Group Margin Total

Creating a Remote Site Setting for Cancelling Orders

When installing or upgrading CME, you can create a remote site setting to support such actions as accessing the Vlocity process Library B2BMobile, canceling orders, and accessing the streaming API. Once you have created the remote site setting, you do not need to do it again for subsequent CME upgrades.

There are three parts that support canceling orders:

- A streaming channel, which is automatically installed.
- A remote site setting, which you must add.
- Interfaces and implementations, specifically:
  - **XOMSupplementalOrderLifeCycle implementation**: Before CME Winter '20, this was the only implementation to handle the full life cycle of a supplemental order when only Cancel was implemented. There was no unified interface to submit a simple order and assetize it.
    For CME Winter '20, the OdinAPIHandler (ODIN) interface was introduced to deal with the life cycle for any kind of order (such as Simple, MACD, or Supplemental) between CPQ and Order Management (OM). In OM Standard, this interface was directly implemented with XOMOMStandardOdinAPIHandler instead of using the default OdinStandardAPIHandler. This release also kept the other implementation, XOMSupplementalOrderLifecycleImpl, to deal with Supplemental order calls during cancellation.
    As of CME Spring '20, only the OMOMStandardOdinAPIHandler is used and XOMSupplementalOrderLifecycleImpl is no longer used. OM Plus uses the standard implementation to deal with extra interfaces, such as SendSubmitToOM, SendFreezeToOM, SendUnfreezeToOM, and OdinSubmitOrderExtraFields.
  - **SupplementalOrderService interface**: This interface is used within the OdinAPIHandler interface, specifically within the OdinAPIAutoAssetizeHandler implementation. When the `preValidate` API or `unfreezeOrder` API call is made from the client, the call uses the implementation configured for the OdinAPIHandler. If OdinAPIAutoAssetizeHandler is the active implementation of the OdinAPIHandler interface, then it internally calls the SupplementalOrderService interface. The only out-of-box implementation for SupplementalOrderService is SupplementalOrderServiceImplementation.
  - **XOMSupplementalOrderLifecycleImpl implementation**: This interface is used from SupplementalOrderServiceImplementation. When the OdinAPIHandler interface is configured with OdinAPIAutoAssetizeHandler as the active implementation, any call to the preValidate API or unfreezeOrder API call is handled through SupplementalOrderService. The only out-of-box implementation for SupplementalOrderService is SupplementalOrderServiceImplementation. This interface (XOMSupplementalOrderLifeCycle) helps SupplementalOrderServiceImplementation freeze and unfreeze orders.
  - **XOMSupplementalOrderLifeCycle interface**: This implementation is for the XOMSupplementalOrderLifeCycle interface. It provides a stub for the freezeOrder and unfreezeOrder API calls. This interface is invoked only when the OdinAPIHandler interface has OdinAPIAutoAssetizeHandler as the active implementation and SupplementalOrderService interface has SupplementalOrderServiceImplementation as the active implementation.

To support canceling orders, you must add your Salesforce org base URL to the remote site settings.

To create a remote site setting:

1. From Setup, in the **Quick Find** box, enter **`Remote Site`**.
2. Click **Remote Site Settings**.
3. Click **New Remote Site**.
4. On the **Remote Site Edit** page, enter the following information:
   - **Remote Site Name** is **`StreamingAPI`**.
   - **Remote Site URL** is the base URL of the CPQ cart. To determine this URL, go to an order and navigate to the CPQ cart, which will have a URL such as `https://cmt-106-vlocity-cmt.na72.visualforce.com/apex/hybridcpq?id=8011H000FEZ84RA5`. It should include

the namespace. The value for the corresponding Remote Site URL is `https://cmt-106-vlocity-cmt.na72.visual.force.com`.
For new Installations, the StreamingAPI should be set as the Remote Site Name.
You need to switch to Salesforce Classic before you invoke the Cart to see that version of the URL.

5. Select **Disable Protocol Security**.

> **IMPORTANT**
> Do not disable protocol security for:
>
> - Streaming API
> - Vlocity Mock
> - Vlocity process Library B2BMobile

6. Select **Active**.
7. Click **Save**.

Required Indexes for Vlocity CPQ Implementations

If you are implementing Vlocity CPQ, add the following required indexes to support the related CPQ objects.

> **NOTE**
> Salesforce does not allow Vlocity to install custom indexes in an org during install. To have a custom index added to an org, open a case with Salesforce technical support and provide the exact SOQL queries with bind variables needed to create and test the index.

If you are implementing Vlocity CPQ, you must add the following required indexes to support the related CPQ objects.

Table 3. Required Indexes for Vlocity CPQ

CPQ Objects	Required Indexes
Opportunity-Based CPQ	• OpportunityLineItem.vlocity_cmt__ParentItemId__c • OpportunityLineItem.vlocity_cmt__RootItemId__c
Order-Based CPQ	• OrderItem.vlocity_cmt__ParentItemId__c • OrderItem.vlocity_cmt__RootItemId__c • OrderItem.vlocity_cmt__AssetReferenceId__c
Quote-Based CPQ	• QuoteLineItem.vlocity_cmt__ParentItemId__c • QuoteLineItem.vlocity_cmt__RootItemId__c • QuoteLineItem.vlocity_cmt__AssetReferenceId__c

CPQ Objects	Required Indexes
Contract and Asset-Based Operations	• Asset.vlocity_cmt__ParentItemId__c • Asset.vlocity_cmt__RootItemId__c • Asset.vlocity_cmt__AssetReferenceId__c • Asset.vlocity_cmt__PricebookEntryId__c • vlocity_cmt__ContractLineItem__c.vlocity_cmt__AssetReferenceId__c • vlocity_cmt__ContractLineItem__c.vlocity_cmt__PricebookEntryId__c

Sample SOQL Queries Requiring Custom Indexes

The following sample SOQL queries require custom indexes. You may need to update the ID values in the WHERE clauses so that they refer to an actual record in your Salesforce org.

Asset SOQL Queries

Asset.vlocity_cmt__ParentItemId__c

```
SELECT
Id,ParentId,vlocity_cmt__AssetReferenceId__c,vlocity_cmt__ParentItemId__c,vloci
ty_cmt__PricebookEntryId__c,vlocity_cmt__RootItemId__c FROM Asset WHERE
vlocity_cmt__ParentItemId__c = '02i3i00000099TNAAY'
```

Asset.vlocity_cmt__RootItemId__c

```
SELECT Id,ParentId,vlocity_cmt__AssetReferenceId__c,vlocity_cmt__RootItemId__c
FROM Asset WHERE vlocity_cmt__RootItemId__c = '02i3i00000099TaAAI'
```

Asset.vlocity_cmt__AssetReferenceId__c

```
SELECT Id,ParentId,vlocity_cmt__AssetReferenceId__c FROM Asset WHERE
vlocity_cmt__AssetReferenceId__c = 'f49e9fa7-2f95-814f-084a-bc017de032ff'
```

Asset.vlocity_cmt__PricebookEntryId__c

```
SELECT
Id,ParentId,vlocity_cmt__AssetReferenceId__c,vlocity_cmt__PricebookEntryId__c,v
locity_cmt__RootItemId__c FROM Asset WHERE vlocity_cmt__PricebookEntryId__c =
'01u3i000000PavqAAC'
```

Order Item SOQL Queries

OrderItem.vlocity_cmt__ParentItemId__c

```
SELECT Id,OrderId,vlocity_cmt__ParentItemId__c FROM OrderItem WHERE
vlocity_cmt__ParentItemId__c = '8023i0000000EBkAAM'
```

OrderItem.vlocity_cmt__RootItemId__c

```
SELECT Id,OrderId,vlocity_cmt__ParentItemId__c,vlocity_cmt__RootItemId__c FROM
OrderItem WHERE vlocity_cmt__RootItemId__c = '8023i0000000EBkAAM'
```

OrderItem.vlocity_cmt__AssetReferenceId__c

```
SELECT
Id,OrderId,vlocity_cmt__AssetReferenceId__c,vlocity_cmt__ParentItemId__c,vlocit
y_cmt__RootItemId__c FROM OrderItem WHERE vlocity_cmt__AssetReferenceId__c =
'f8c87dd8-42bd-acce-c064-0ff505aa0883'
```

Editing $root.settings and $root.vlocityCPQ.features Variables

In Winter '18, the UI settings for Vlocity CPQ have been moved from `rootScope` to a service. If you have custom templates that use `$root.settings` or `$root.vlocityCPQ.features` variables, you must edit the following files:

- cpq-base-grid.html
- cpq-base-header.html
- cpq-base-header-card.html
- cpq-left-sidebar.html
- cpq-product-cart.html
- mobile-ret-cart.html

Take These Files	Make These Changes

cpq-base-grid.html
 	
mobile-ret-cart.html
	• Delete the `initApiSettingsController` from the JavaScript section.
	• Delete references to the `initApiSettingsController` controller; for example:
	
	vlocity.cardframework.registerModule.controller('initApiSettingsController', ['$rootScope', function($rootScope) {
	
	$rootScope.apiSettings = {
	
	'addToCartAPIRequiresPricing': true,
	
	'addToCartAPIRequiresValidation': true,
	
	...
	
	};
	
	$rootScope.vlocityCPQ.features = {
	
	enablePromotions : true,
	
	...
	
	};
	
	}]);

cpq-base-header-card.html	Replace `$root.vlocityCPQ.features.enablePricing` with `attrs.enablePricing`.
cpq-base-header.html	• Update the `vloc-card` directive. • Add the `enable-pricing` parameter: <vloc-card enable-pricing="attrs.enablePricing">
cpq-left-sidebar.html cpq-product-cart.html:	Replace `$root.vlocityCPQ.features` with `importedScope.featureSettings`.

GUIDs Performance

Vlocity made GUID performance enhancements to meet demand from customers for higher throughput for some key transaction types. The GUID performance feature is an optional performance enhancement available to customers with the Fall '20 release.

GUIDs Performance in Hierarchy

Vlocity made GUID performance enhancements to meet demand from customers for higher throughput for some key transaction types, allowing for the processing of 100,000 orders per hour.

The transaction types can be a mix, but a large portion of that mix is add-to-cart, which essentially adds line items to an existing or empty order. To achieve the desired throughput, the number of, and time spent on, DMLs and SOQLs executed by the add-to-cart transaction in the modeled customer environment has been reduced.

Of the two types of database operations, DMLs are typically far fewer in number but are often very slow to execute and expensive in terms of database resources. In particular, the DML operations are expensive in terms of a metric tracked closely by Salesforce: Database CPU. Salesforce requires that none of their pods exceed 40% DB CPU utilization. The combination of the high number of order transactions in the performance tests modeled on the customer requirement and the impact of each transaction was easily exceeding that limit.

The add-to-cart transaction issues three DMLs:

- Insert DML, required to create the line items added to the cart.
- DML for pricing. This DML was eliminated by performing the pricing in memory before the initial insert.
- DML to update the hierarchy establishing fields ParentItemId__c and RootItemId__c that contain the string-representations of Id values of related records.
  Since the ParentItemId__c and RootItemId__c fields contain ID values of rows that are often not created until after the first DML executes, you cannot eliminate the second DML while maintaining the existing data model where those fields contain, essentially, Salesforce ID values.
  However, the actual type of the ParentItemId__c and RootItemId__c fields is text with a maximum length of 255 characters. This type is capable of storing both string representations of Salesforce ID values but can also store any other kind of string. Therefore, instead of storing the string representation of the Salesforce ID of the target row, the target is identified by its AssetReferenceId__c value so the same result can be accomplished without requiring this DML.

Previous Data Model for Hierarchy-Establishing Fields

In all releases since CME Summer ‘19, the ParentItemId__c and RootItemId__c fields of the line items contained Salesforce IDs. The line item types are OrderItem, OpportunityLineItem, and QuoteLineItem, all of which have the two fields defined by the Vlocity package. In addition, the Asset type functions much like line items in many cases and has the same two fields for establishing the hierarchy.

In the previous data model, the ParentItemId__c and RootItemId__c fields contain the Salesforce ID values of the target.

As an example, consider a simple product hierarchy with a root product named "Root Product" and a single child named "Child Product". If "Root Product" were added to an order, the OrderItems would contain the following fields, in addition to many others not shown:

ID	AssetReferenceId__c	Name	ParentItemId__c	RootItemId__c
8024600000Nj76BAAR	77be5715-a534-cd6f-d634-7a23518de2be	Root Product	NULL	8024600000Nj76BAAR
8024p00000NDeLWAA1	29465e8f-6521-794b-0f66-2ce1d93d4991	Child Product	8024600000Nj76BAAR	8024600000Nj76BAAR

To represent the hierarchy, the ParentItemId__c and RootItemId__c fields of "Child Product" contain the Id-value of "Root Product" (8024600000Nj76BAAR). In addition, Root Product's RootItemId__c contains its own ID value because it is the root of the hierarchy.

New Data Model for Hierarchy-Establishing Fields

In the new data model, the ParentItemId__c and RootItemId__c fields contain the AssetReferenceId__c value of the target row.

For example, using the new data model the OrderItem rows are:

ID	AssetReferenceId__c	Name	ParentItemId__c	RootItemId__c
8024600000Nj76BAAR	77be5715-a534-cd6f-d634-7a23518de2be	Root Product	NULL	77be5715-a534-cd6f-d634-7a23518de2be
8024p00000NDeLWAA1	29465e8f-6521-794b-0f66-2ce1d93d4991	Child Product	77be5715-a534-cd6f-d634-7a23518de2be	77be5715-a534-cd6f-d634-7a23518de2be

The hierarchy-establishing fields contain the AssetReferenceId__c value of the target rather than the ID value of the target. You can tell immediately that GUID values are used because the ParentItemId__c and RootItemId__c values are longer than normal Salesforce IDs and contain dashes. This can be an important clue for debugging and testing or while searching data for errors.

However, while AssetReferenceId__c values often look like the values above, they may also look different and even take the form of Salesforce IDs. The values above are generated by default Vlocity code.

You can implement an open interface to generate your own asset reference ID, in which case the values could look like any string. Finally, it is possible that the values in the AssetReferenceId__c field are taken from older Vlocity code that actually used Salesforce ID values. Generally, Salesforce ID values come from another container, such as values that look like Salesforce IDs but don't match the IDs of other line items in the current container and instead match the AssetReferenceId__c values.

Configuration

Due to the heavy testing requirements of these changes and the potential for customer-logic incompatibility, the ability to use GUID values in the hierarchy-establishing fields is hidden behind a master configuration switch. By default, the switch is logically off, meaning that Salesforce ID values are still used in the hierarchy fields.

In CME Fall ’20, you can enable the GUID mode by adding a CPQ configuration setting named `UseAssetReferenceIdForParentAndRoot` and setting its value to **True**.

Once this configuration setting is enabled, Vlocity CME code immediately starts producing new records, such as OrderItem, OpportunityLineItems, QuoteLineItems, and Assets, with the GUID values in the hierarchy fields when conditions allow for it, such as when creating a new order.

These new records are not compatible with code that runs when the configuration switch is turned off, so you cannot turn off the feature again without potential data-incompatibility errors.

Impact of GUID-Capable Data Model Change

The GUID-capable data model change has a wide impact, especially regarding the code change for a variety of transaction types that assumed the values stored in these fields were Salesforce IDs. Over 40 Vlocity via_telco Apex modules had non-trivial logic with the assumption of Id-values in these fields built-in. All were modified to allow for the possibility that these fields may contain GUIDs when the master configuration switch is on.

Also, customers still reap performance benefits for non-legacy containers, such as new orders, even if they have legacy data, such as unconverted assets.

All Salesforce Vlocity code must be capable of handling legacy data when the switch is on even if you have a data migration strategy.

Customer Migration Strategy and Legacy Data

Once the master GUID switch is enabled, the target org immediately recognizes the performance benefits of using GUIDs when operating on empty containers, such as when creating a new order. However, existing data, such as existing OrderItems and Assets, cannot be converted instantly. Since no program running in the background can convert all data instantly, and since downtime while any such program is running is generally unacceptable, all the GUID changes have been implemented so they are backward compatible with existing non-GUID records.

To simplify the implementation and allow for high performance, containers, such as Order, Opportunity, Quote, or Account, do not contain a mix of records with both GUID and non-GUID values in the ParentItemId__c and RootItemId__c fields. This allows the bulk of the Vlocity code that implements the various transaction types to determine which type of hierarchy-establishing value to use for the current container by examining exactly one field in one existing record in that container. This field is usually a RootItemId__c field because null values are not normally used in this field.

If there are existing records, then any new records will match the style of the existing records in the container. For example, if an OrderItem is added to an existing Order with the master-GUID configuration switched on and that order has OrderItem records containing the old Salesforce IDs in the hierarchy fields, then the new OrderItem will also have Salesforce IDs.

OrderItems added to an Order that has existing OrderItems with GUIDs will also use GUIDs. The related add-to-cart transaction incurs performance benefits as a result. OrderItems added to an empty order will have GUIDs because there is no risk of mixing.

An Apex batch process converts Vlocity-specific customer data in the background to use the GUIDs. This program is compatible with the running Vlocity Apex code on the same org by using DMLs that convert one or more containers at a time, but never partial containers.

Customers who enable this performance feature may also have custom logic to convert for maximum performance benefit. For example, some customers may produce orders by importing them continually from a third-party order management system. To work efficiently with the new GUID change, such a customer would need to modify that logic so the new orders are produced with the appropriate GUID values in the ParentItemId__c and RootItemId__c fields. If the customer did not change the logic, the orders produced would still work by running Vlocity code even with the master GUID switch enabled. However, any transactions operating on those orders would not incur performance benefits.

GUIDs Performance in Vlocity CPQ

The GUID performance feature is an optional performance enhancement available to customers with the Fall '20 release. When enabled, the feature eliminates an expensive write operation from many common customer transactions, such as add-to-cart.

The existence of an ID value within the RootItemId__c and ParentItemId__c fields in the without-GUIDS case usually requires a separate DML to write after the DML that created the line items. The reason for the additional DML is that the ID values of the target line items are usually not available until after the write operation that created them is finished. This second write is not necessary when you enable the GUIDs performance feature unless you are creating new line items within containers that have not been converted to the new style because the new values are not Salesforce IDs.

In addition to immediate time savings, the GUIDs performance feature carries other benefits for the future, allowing existing features and to-be-developed features to potentially run faster by enabling them to issue minimal write operations (one). Also, the savings of the DML can have cascading benefits for other database operations on the same Salesforce org by reducing the overall database workload.

DMLs can be very expensive to execute so even though this change does not come without potential migration issues, the benefit is substantial both in immediate time savings and reducing the load on the database in general.

It is not possible to accurately quantify the time savings enabled by this feature because observed times for the affected write operations vary widely depending on factors such as the exact structure of the product being added, load on the Salesforce org, and many other conditions.

Enable GUIDs

NOTE

Do not turn on this setting unless a review of the existing custom code has been performed.

To enable the GUID performance feature:

1. From the App Launcher, click **CMT Administration**.
2. Click **CPQ Configuration Setup**.
3. Click the Edit icon for **UseAssetReferenceIdForParentAndRoot**.
4. Change the **Setup Value** to **True**.

Impact of Enabling GUIDs

Enabling the GUIDs feature impacts the following line item and asset types stored in the RootItemId__c and ParentItemId__c Vlocity-defined fields:

- Asset contained within an Account
- OrderItem contained within an Order
- OpportunityLineItem contained within an Opportunity
- QuoteLineItem contained within a Quote

For example, consider a simple product hierarchy with a root product named "Root Product" and a single child named "Child Product. The line items, which could be OrderItems within an Order container, but in fact could be any of the above types, might appear in the database without the GUID switch enabled:

ID	AssetReferenceId__c	Name	ParentItemId__c	RootItemId__c
8024600000Nj76BAAR	77be5715-a534-cd6f-d634-7a23518de2be	Root Product	NULL	8024600000Nj76BAAR
8024p00000NDeLWAA1	29465e8f-6521-794b-0f66-2ce1d93d4991	Child Product	8024600000Nj76BAAR	8024600000Nj76BAAR

These are not all the fields present, but these are all the fields relevant to the GUID feature changes. The ParentItemId__c values, when not null, and the RootItemIdValues__c all match the target Salesforce ID (“Id”) of some row in the container. The ParentItemId__c points to the ID of the immediate parent in the hierarchy and the RootItemId__c field points to the line item at the root of the hierarchy. The AssetReferenceId__c field value is not entirely relevant at this point, but it becomes relevant as you explore the with-GUID state of the records.

Consider the state of these same records as they would exist if written into an empty container with the GUID feature enabled:

ID	AssetReferenceId__c	Name	ParentItemId__c	RootItemId__c
8024600000Nj76BAAR	77be5715-a534-cd6f-d634-7a23518de2be	Root Product	NULL	77be5715-a534-cd6f-d634-7a23518de2be
8024p00000NDeLWAA1	29465e8f-6521-794b-0f66-2ce1d93d4991	Child Product	77be5715-a534-cd6f-d634-7a23518de2be	77be5715-a534-cd6f-d634-7a23518de2be

Note that the ParentItemId__c and RootItemId__c values no longer match the target line item’s ID field but instead match the target line items’ AssetReferenceId__c field. The hierarchy defined is the same in both cases, without GUID, and with GUID, but the type of value within the hierarchy fields is different.

Also, note that neither the GUID nor without GUID examples mix the value styles. There is not, for instance, one-row using Salesforce IDs and another using AssetReferenceId__c values in the two fields. Line items or assets in a single container do not mix styles even with the GUID switch on.

The ParentItemId__c and RootItemId__c fields point to the AssetReferenceId__c value of the relevant line item in the hierarchy instead of the Salesforce ID of those line items.

Handling of Legacy Records

You do not have to migrate legacy records in the customer database that contain values written before you enabled the GUID switch because those records already have proper functionality. However, if you proactively change the records to the new style, more types of transactions, including MACDs writing to older-style accounts with older-style assets, experience the performance benefits of the GUID feature. Salesforce Vlocity provides customers with a background job capable of making these changes proactively.

Once you enable the GUID switch, existing containers are not immediately converted to the new data format. Instead, Vlocity code can detect and handle either of the two formats to allow proper functionality while existing data is migrated using the supplied background process (or not, the migration is optional).

Custom Code

Ensure all such custom code is inspected and modified before enabling the GUID feature. It is possible that a customer installation has code that is sensitive to this change.

GUIDs Performance - Inspect and Modify Custom Code

You can inspect and adapt custom Apex code to handle the new values present in the ParentItemId__c and RootItemId__c fields of the Asset, OrderItem, OpportunityLineItem, and QuoteLineItem SObjects.

Recommended Approach for Identifying Problem Areas

Vlocity recommends you do a code inspection before testing to identify problems in custom-code. Issues regarding custom-code sensitivity to the data model changes are not always readily apparent and may pass a cursory test.

Before You Begin
1. Ensure you are familiar with the GUIDs Performance in Vlocity CPQ. 2. Have a general understanding of the feature and familiarity with Apex coding on the Salesforce platform

To perform the code inspection:

1. Search the custom code for keywords ParentItemId and RootItemId.
   - Watch for search results that may occur in variable names or within SOQL queries and may include the suffix `'__c'`, depending on the context. Examine all such results and flag them if they relate to the value stored in either a `ParentItemId__c` field or `RootItemid__c` field.
   - If there are no search results, then conduct thorough testing. However, if no results were found, then it's likely that no changes are necessary.
2. For search results found in step 1, determine if the result is for a SOQL statement. For references within SOQL statements, look for the following:

- Is the reference to `vlocity_cmt__RootItemId__c` or `vlocity_cmt__ParentItemId__c` fields one of the affected types (Asset, OrderItem, QuoteLineItem or OpportunityLineItem)? If not, you can safely remove it from consideration.
- Is the check for null or not null only? If so, you can safely remove it from consideration.
- If the checks above do not remove the search result from consideration, then you must modify the result. See the next section.

3. For hits found in step 1, determine if the hit is within non-SOQL Apex code. For references within non-SOQL Apex code, look for the following:
   - Is the reference to the `vlocity_cmt__RootItemId__c` or `vlocity_cmt__ParentItemId__c` field value of one of the affected SObject types (Asset, OrderItem, QuoteLineItem or OpportunityLineItem)? If not, you can safely remove it from consideration.
   - Is the reference only to compare the value for null or not-null? If so, you can safely remove it from consideration.
   - If the checks above don't remove the search result from consideration, then you must modify the result. See the next section.
4. After inspecting and correcting the custom code, test the code to confirm proper operation.

Guidelines for Adapting References to RootItemId__c and ParentItemId__c

If the checks described in the previous section identify areas that need modification, then you need to adjust the custom code. The specific cases that need adjusting are those in which the values taken from the `RootItemId__c` and `ParentItemd__c` fields are treated as Salesforce IDs. If the values are treated strictly as strings and never compared to any other Salesforce ID value, then the code may still execute without modification.

However, if the specific case has passed the tests in the previous section then it likely identifies code that needs modification.

Here are some additional tests that can confirm the if code needs modification:

- A value taken from `RootItemId__c` or `ParentItemId__c` is cast to a Salesforce ID value, for example, `Id rootId = (Id) someChildSObject.get('vlocity_cmt__RootItemId__c')`.
- A value taken from `RootItemId__c` or `ParentItemId__c` is referenced in a SOQL statement where it is compared to other Salesforce ID values, for example, `String parentProductName = [SELECT Product2.Name FROM OrderItem WHERE Id = :myChildOrderItem.vlocity_cmt__ParentItemId__c];`.

If the following condition is true, then, despite passing the checks in the previous section, the custom code may not need modification.

The value taken from `RootItemId__c` or `ParentItemId__c` field is treated only as a string value and is never compared to a Salesforce ID. For example, a value is fetched from one of these fields and used in a map as a unique key field of type String. The keys in the map are compared only to other `RootItemId__c` and `ParentItemId__c` fields in the same container and never mapped to nor compared with Salesforce ID values.

Once you have identified custom code that must be changed, remember to maintain compatibility with earlier releases in the modified logic.

Maintain Compatibility with Previous Releases

The ParentItemId__c and RootItemId__c fields are populated for new records created when the master GUID switch is enabled. However, there is no way to instantaneously convert all existing records. Therefore, even if your organization intends to convert all legacy records using the provided conversion procedure, you should allow for the possibility that your custom code could be operating on legacy records that have not been converted.

This means, in practice, a given container, such as Account, Quote, Order, or Opportunity, can contain either updated records using AssetReferenceId__c values in ParentItemId__c and RootItemId__c (new records) or Salesforce IDs in those fields (legacy records). There can never be a mix of such types within the same container, so you only need to test one record to determine the style that is in effect if you've retrieved the line items by reference to the container ID (i.e., selected all assets from an account).

Identifying the ParentItemId__c and RootItemId__c style for the current container

As noted in the previous section, any given container could be using AssetReferenceId__c values or Salesforce ID values within the ParentItemId__c and RootItemId__c fields of the contained line items. In general, custom code that needs to be modified will need to determine which type of style is in use to successfully navigate the hierarchy.

Here is the recommended way to check for which style is in use:

1. Obtain a root item with at least the Id, AssetReferenceId__c, and RootItemId__c fields populated.
   - You can identify a root line item by testing for ParentItemId__c = null.
   - Your custom code may have already fetched or otherwise obtained a root line item. If so, it's recommended for performance reasons to leverage the existing sobject if possible rather than perform a separate fetch for this purpose.
2. Compare the ID value as a string to the RootItemId__c value. If they are the same, then the current container is using legacy Salesforce IDs. Otherwise, the container is using the newer AssetReferenceId__c values.

Here is an example of code that, given a list of sobjects from a single container, can identify which value style is in use. A requirement of the input list is that at least one root line item must be present. This is guaranteed if, for instance, the line items represent the complete contents of the container. If there was an error detected in the input sobject list, the code returns null. Otherwise, it returns true if the container is using AssetReferenceId__c values and false if not using AssetReferenceId__c values. The example makes use of a second method that accepts a root line item as input. This can also be useful for special cases where the root line item has already been identified.

```
public Boolean isContainerUsingGuids(List<SObject> lineItemList)
{
    SObject rootLineItem = null;
    for (SObject currentLineItem : lineItemList)
    {
        if
(currentLineItem.get('vlocity_cmt__ParentItemId__c')
```

```
             == null)
        {
            rootLineItem = currentLineItem;
            break;
        }
    }
    if (rootLineItem == null)
    {
        // error - no root item supplied
        return null;
    }
    return
isRootLineItemUsingGuids(rootLineItem);
}
public Boolean isRootLineItemUsingGuids(SObject rootLineItem)
{
    String rootIdValueAsString = (String) rootLineItem.Id;
    if (rootIdValueAsString == null)
    {
        // error - the id field is not opulated.
        return null;
    }
    String rootRootItemIdAsString =
        (String) rootLineItem.get('vlocity_cmt__RootItemId__c');
    return (rootIdValueAsString != rootRootItemIdAsString);
}
```

Tips for Modifying Custom Code for Specific Cases

It is not possible to enumerate all the use cases and adjustments here for the new values found in the ParentItemId__c and RootItemId__c fields. This section lists some of the more likely scenarios and suggests strategies for adjustment.

Case 1: ParentItemId__c fetches the parent SObject

Here is an example of such a query as it may exist in your current code:

```
OpportunityLineItem parentOppLineItem = [
    SELECT Id, UnitPrice, Product2.Name
      FROM OpportunityLineItem
      WHERE Id=:childOppLineItem.vlocity_cmt__ParentItemId__c];
```

You can adjust this example to work with GUIDs, as follows:

```
OpportunityLineItem parentOppLineItem;
if (containerIsUsingGuids)
{
    // using guids, ParentItemId__c is comparable to
```

```
    // AssetReferenceId__c
    parentOppLineItem = [
        SELECT Id, UnitPrice, Product2.Name
          FROM OpportunityLineItem
         WHERE vlocity_cmt__AssetReferenceId__c =
         :childOppLineItem.vlocity_cmt__ParentItemId__c];
}
else
{
    // using legacy salesforce ids. ParentItemId__c is comparable to Id.
    parentOppLineItem = [
        SELECT Id, UnitPrice, Product2.Name
          FROM OpportunityLineItem
         WHERE Id=:childOppLineItem.vlocity_cmt__ParentItemId__c];
}
```

This example assumes that the `containerIsUsingGuids` value is determined by logic similar to the `isContainerUsingGuids()` method supplied above.

Case 2: RootItemId__c identifies line items within a hierarchy having the same root

RootItemId__c is used as input to the query for selecting the line items in the hierarchy, but that case is similar enough to the previous case (using ParentItemId__c) that it's not repeated here. Instead, consider the case where all line items in the container have already been fetched into memory and the RootItemId__c value collects the line items belonging to a single hierarchy.

The existing code to collect the hierarchy might look something like the following example, which assumes that a previously identified rootAsset and a list of all assets in the container are supplied as input.

```
List<Asset> allAssetsInHierarchy = new List<Asset>();
for (Asset currentAsset : allAssetsInContainer)
{
    if (currentAsset.vlocity_cmt_RootItemId__c == rootAsset.Id)
    {
        allAssetsInHierarchy.add(currentAsset);
    }
}
```

The code identifies the assets in the hierarchy by comparing the RootItemId__c value to the Salesforce ID of the known root. This does not always work when the GUID feature is enabled.

Here is how the code might look after adjusting for the GUID performance feature. It has the same input assumptions and leverages the isRootLineItemUsingGuids method defined above.

```
Boolean usingGUIDs = isRootLineItemUsingGuids(rootAsset);
List<Asset> allAssetsInHierarchy = new List<Asset>();
for (Asset currentAsset : allAssetsInContainer)
```

```
{
    if (usingGUIDs)
    {
        if (currentAsset.vlocity_cmt__RootItemId__c ==
            rootAsset.vlocity_cmt__AssetReferenceId__c)
        {
           allAssetsInHierarchy.add(currentAsset);
        }
    }
    else
    {
        if (currentAsset.vlocity_cmt_RootItemId__c ==
            rootAsset.Id)
        {
            allAssetsInHierarchy.add(currentAsset);
        }
    }
}
```

GUIDs Performance with Data Migration

You can use the GUIDs Performance feature for converting affected records in the interest of obtaining maximum performance benefit. This allows you to maximize the performance of certain operations when they interact with the affected records.

This process is optional. There is no need to convert records using this process to obtain proper functionality with the GUID feature on or off.

The GUIDs performance feature is built to be compatible with earlier releases. Errors do not occur if legacy records are not converted. However, if legacy records are not converted, then an extra write DML operation may occur during transactions that creates additional records in the same container. A container can be an:

- Account containing assets
- Order containing OrderItems
- Quote containing QuoteLineItems
- Opportunity containing OpportunityLineItems

For example, if legacy assets are not converted and the GUID feature is enabled, performance benefits are still the same for any resulting Asset-to-Order operation because, for example, the order is entirely new even though it was modeled after the existing unconverted Assets. The new order line items in the new order can use GUIDs without violating the mixing rule and obtain improved performance as a result.

However, if one or more order items are added to the order and the order is submitted, the resulting submit operation incurs an extra, potentially costly, DML write operation. The reason is that the relevant target container in this operation is the existing Account that already has unconverted assets. To avoid violating the mixing principle, the code must then write the new assets using Salesforce IDs, which requires a separate write operation.

Running the Conversion Batch Process

Salesforce Vlocity provides unmanaged Apex code to facilitate the one-time conversion of legacy records. The conversion process leverages Salesforce's batch processing. For more information on running batch apex, see Salesforce documentation.

The only fields that are modified are the vlocity_cmt__ParentItemId__c and vlocity_cmt__RootItemId__c fields of the related line items: Asset, OrderItem, QuoteLineItem, and OpportunityLineItem. You can inspect the code before running it.

The name of the batch class is `GUIDConverterBatchProcessor`. This class is designed to either identify or convert needed records of a single container type within each scheduled run. The primary output interface is email, so ensure the email address of the user account for initiating the batch process is up to date. Check your spam filter as well.

Initiate Immediate Batch Processing

Before You Begin
Install the GUIDConverterBatchProcessor.cls on the target org.

To initiate immediate batch processing:

1. Execute code to create an instance of GUIDConverterBatchProcessor.cls and schedule the batch job. Each such instance is specific to one container type. For example, to start a job for immediate execution, subject to org resources, to convert all assets tied to any account on the org that does not already conform to the more efficient data model, run this in anonymous Apex:

```
GUIDConverterBatchProcessor
converter =
     new GUIDConverterBatchProcessor(
         'Account',
GUIDConverterBatchProcessor.Options.CONVERT,
                null);
     Id jobId =
Database.executeBatch(converter);
     System.debug(LoggingLevel.ERROR, 'job id
is: ' + jobId);
```

2. Wait for the job to finish and look for an email sent to the email address of the Salesforce user account for launching the batch job. The email details the number of containers (accounts in this case) for which conversion was necessary and lists the account IDs of any accounts that could not be processed.

Here are the details about the conversion launching code used in the above example:

- This example converts only assets. To completely cover all possible records, run a similar process for Order, Quote, and Opportunity, using each as the first parameter.
- This example converts the affected records. If you would rather get an indication of how many records might need to be modified, you can use other options for the second parameter besides CONVERT.

- The third parameter is null in this example. Use null to specify containers to skip if errors occurred in previous runs.

Options for GUIDConverterBatchProcessor

The options for the GUIDConverterBatchProcessor are supplied through the constructor, as in the example above. These options are:

- **Parameter #1: Container Name**
  This parameter can be one of the following:
  - Account -- for Asset scanning or conversion.
  - Order -- for OrderItem scanning or conversion.
  - Quote -- for QuoteLineItem scanning / conversion.
  - Opportunity -- for OpportunityLineItem scanning / conversion.

  If Account is specified then Assets are scanned or converted, depending on the option you chose in the second parameter, for each Account in the system. Similarly, if Order is specified, then OrderItem records are scanned, etc.
- **Parameter #2: Option**
  This parameter can be any of the values specified in the GUIDConverterBatchProcessor.Options enum:
  - COUNT_ONLY
    - Scans the record types indicated by the first parameter.
    - Returns a count of the number of containers that need conversion in the response email.
    - Does not modify data.
  - SUPPLY_IDS_ONLY
    - Scans the record types indicated by the first parameter.
    - Returns a count of the number of containers that need conversion and specifies the Salesforce ID values of those containers in the response email.
    - Does not modify data.
  - CONVERT
    - Scans and converts the record types indicated by the first parameter.
    - Returns a count of the containers converted and a list of the Salesforce IDs of any container that could not be converted due to an error condition.
    - If any containers require conversion, related hierarchy fields `vlocity_cmt__ParentItemId__c` and `vlocity_cmt__RootItemId__c` are modified in the related line item records.
- **Parameter #3: Container Skip List**
  This parameter is optional. It can be null. If specified, this parameter must be a list of containers IDs that should not be scanned or converted during the run. The intended use is null unless a problematic container, such as Account, was identified during a previous batch job. Such containers are identified by ID in the response email and can be supplied to the constructor so they do not affect the processing of other containers.

Governor Limits and Need for Limiting Batch Size

Apex batch job execution is subject to governor limits that are similar to those for other transaction types for each batch execution. For the GUIDConverterBatchProcessor, a batch is a number of containers specified by ID. That batch size is controlled by an optional second parameter to the Database.executeBatch() call but defaults to 200.

If you encounter governor limit exceptions while executing the batch process, try reducing the batch size by passing, for example, 100 as the second parameter to Database.executeBatch()).

Picklist Value Activation

During a package upgrade, picklist values are not normally installed into your org for existing fields. As a result, you must manually add the following picklist values. However, if any picklist values for an object are installed through the Vlocity package and those picklist values are inactive, you must activate them. Also, for several picklist values, you must remove them and either change the Lookup values to `text` or leave them blank.

Ensure the following picklist values are added, activated, or removed:

Object Label and API Name	Field Label and Field Name	Picklist Value to Add
Context Scope `vlocity_cmt__ContextScope__c`	Scope Type `vlocity_cmt__ScopeType__c`	Virtual
Vlocity OmniScript Element `vlocity_cmt__Element__c`	Type `vlocity_cmt__Type__c`	Radio Group Delete Action Navigate Action Batch Action Cache Block Try Catch Block Conditional Block Custom Lightning Web Component
Vlocity OmniScript `vlocity_cmt__OmniScript__c`	Language `vlocity_cmt__Language__c`	Multi-Language
Product `Product2`	Scope `vlocity_cmt__Scope__c`	Account Order Item Downstream Order Item Top Order Item
Product `Product2`	Product Family `Family`	Multiplay Mobile
Product `Product2`	Lifecycle Status `vlocity_cmt__LifecycleStatus__c`	Add • Released • Amendment Pending Remove • Completed • Deployed

Object Label and API Name	Field Label and Field Name	Picklist Value to Add
Product	Type	Plan
`Product2`	`vlocity_cmt__Type__c`	Service
		SIM Card
		Add On
		Fee
		Top Up
		Handset
Product	Sub Type	Postpaid
`Product2`	vlocity_cmt__SubType__c	Prepaid
Pricing Variable	Scope	Parent
`vlocity_cmt__PricingVariable__c`	`vlocity_cmt__Scope__c`	Parent Rollup
		Group
		Group Rollup
		Master
		Master Rollup
Order Product Relationship	Relationship Type	Replacement
`vlocity_cmt__OrderItemRelationship__c`	`vlocity_cmt__RelationshipType__c`	Reassignment
		Movement
Asset Relationship	Relationship Type	Replacement
`AssetRelationship`	`RelationshipType`	Reassignment
		Suspend
		Resume
		Movement
		Relies On
Order Product	Sub Action	Replace
`OrderItem`	`vlocity_cmt__SubAction__c`	Reassign
Order Product	Fulfilment Status	Rejected
`OrderItem`	`vlocity_cmt__FulfilmentStatus__c`	
Quote Line Item	Action	Suspend
`QuoteLineItem`	`vlocity_cmt__Action__c`	Resume
Quote Line Item	Sub Action	Replace
`QuoteLineItem`	`vlocity_cmt__SubAction__c`	Reassign

Object Label and API Name	Field Label and Field Name	Picklist Value to Add
cachedAPIResponse `vlocity_cmt__cachedAPIResponse__c`	Type `vlocity_cmt__Type__c`	ruleSetCombinationResultItem catalogCtxQualificationMap groupContext basketWrapper assetReferenceKey pleOfferMap basketPricing bundleWrapper promotionWrapper
CachedAPIChange `vlocity_cmt__CachedAPIChange__c`	ChangeType `vlocity_cmt__ChangeType__c`	promotionWrapper
Order Pricing `vlocity_cmt__OrderPriceAdjustment__c`	Source `vlocity_cmt__Source__c`	Attribute Based Pricing
Quote Pricing `vlocity_cmt__QuotePricingAdjustment__c`	Source `vlocity_cmt__Source__c`	Attribute Based Pricing
Opportunity Pricing `vlocity_cmt__OpportunityPriceAdjustment__c`	Source `vlocity_cmt__Source__c`	Attribute Based Pricing
Account Pricing `vlocity_cmt__AccountPriceAdjustment__c`	Source `vlocity_cmt__Source__c`	Attribute Based Pricing
Account Hold `vlocity_cmt__AccountHold__c`	Hold Type `vlocity_cmt__HoldType__c`	Customer Initiated Company Initiated Fraud
Asset `Asset`	Action `vlocity_cmt__Action__c`	Suspend Resume
Opportunity Product `OpportunityLineItem`	Action `vlocity_cmt__Action__c`	Suspend Resume
Opportunity Product Relationship `vlocity_cmt__OpportunityLineItemRelationship__c`	Relationship Type `vlocity_cmt__RelationshipType__c`	Replacement Movement Reassignment
Quote Product Relationship `vlocity_cmt__QuoteLineItemRelationship__c`	Relationship Type `vlocity_cmt__RelationshipType__c`	Movement

Object Label and API Name	Field Label and Field Name	Picklist Value to Add
Offer Migration Plan `vlocity_cmt__OfferMigrationPlan__c`	Product Family `vlocity_cmt__ProductFamily__c`	Broadband Fixed Line TV Wireless Mobile Internet Multiplay
Product Relationship `vlocity_cmt__ProductRelationship__c`	Relationship Type `vlocity_cmt__RelationshipType__c`	Relies On
Order `Order`	Order Status `vlocity_cmt__OrderStatus__c`	Amend Requested Cancel In Progress Superseded Submitted Rejected Frozen Discarded Available In Pool
Order `Order`	Fulfilment Status `vlocity_cmt__FulfilmentStatus__c`	Queued Rejected
Order `Order`	Order Type `Type`	Sales Suspend Lost or Stolen SIM Resume Lost or Stolen SIM Replace SIM Change Number
Orchestration Scenario `vlocity_cmt__OrchestrationScenario_c`	Action `vlocity_cmt__Action_c`	NoChange Suspend Resume
Vlocity Object or Object Type `vlocity_cmt__ObjectClass__c`	Lifecycle Status `vlocity_cmt__LifecycleStatus__c`	Add: • Released Remove: • Completed • Deployed

Object Label and API Name	Field Label and Field Name	Picklist Value to Add
Vlocity Picklist `vlocity_cmt__Picklist__c`	Lifecycle Status `vlocity_cmt__LifecycleStatus__c`	Add: • Released Remove: • Completed • Deployed
Project `vlocity_cmt__Project__c`	Status `vlocity_cmt__Status__c`	Add: • In-Review • In-Test • Released • Canceled Remove: • Deployed • Completed • Testing • Approved • Rejected

Remove the following picklist value and change the Lookup value to `text` or leave it blank:

- Object Label: Vlocity Attribute
- Object: Attribute__c
- Field: ValueType__c
- Picklist Value: Lookup

Adding Picklist Values

During a package upgrade, picklist values cannot be installed into your org for existing fields. As a result, after an upgrade, you must manually add the new picklist values specified in the post-upgrade steps.

To add values to a picklist:

In Lightning Experience:

1. Go to **Setup** → **Object Manager**, and find the object that contains the picklist field that you need to change.
2. Click **Fields & Relationships**.
3. Click the picklist field.
4. Under the **Values** section, click **New**.
5. Enter the new picklist values.
6. To apply the new values to a specific record type, check the box next to the **Record Type Name**. If no **Record Type Name** boxes are checked, the values apply to all record types.
7. Click **Save**.

In Salesforce Classic (Aloha) interface:

1. Go to **Setup**.
2. Find the object that contains the picklist field that you need to change, either under **Build** → **Customize** or **Build** → **Create** → **Objects**.
3. In **Fields**, click the picklist field.
4. Under the **Values** section, click **New**.
5. Enter the new picklist values.
6. To apply the new values to a specific record type, check the box next to the **Record Type Name**. If no **Record Type Name** boxes are checked, the values apply to all record types.
7. Click **Save**.

API Changes for Picklist and Multipicklist Attributes

If you have custom implementations that use the API response of a Product's picklist attribute values, multipicklist attribute values, or user values, you must adjust these implementations to handle the new response format. The response exists under the productAttributes node of the attributeCategories node. The user-selected values will reflect the value mapped in the list of values.

The changes are as follows:

- Id is mapped to Id
- value is mapped to value (only if value is not null)
- label is mapped to value (only if value is null and label is not null)
- Id is mapped to value (only if value and label are null)
- sequence is mapped to display sequence
- label is mapped to label

Example 1. Modified Attribute API Response Example

```
{
 "code": "Picklist Attribute 2",
 "dataType": "text",
 "inputType": "dropdown",
 "multiselect": false,
 "required": false,
 "readonly": false,
 "disabled": false,
 "filterable": true,
 "attributeId": "a0Y1N00000BomDMUAZ",
 "label": "Picklist Attribute 1",
 "displaySequence": 1,
 "hidden": false,
 "cloneable": true,
 "isNotTranslatable": false,
 "values": [
   {
     "id": "bd8ffecb-aaaf-b060-b5bf-068d1335baeb",
```

```
      "label": "Value 1 label",
      "readonly": false,
      "disabled": false,
      "value": "Value 1",
      "displaySequence": 10
    },
    {
      "id": "4bfff376-d1b3-fb9b-6be5-7b3dd3897ed2",
      "label": "Value 2 label",
      "readonly": false,
      "disabled": false,
      "value": "Value 2",
      "displaySequence": 20
    },
  ],
  "userValues": "Value 2"
}
```


You can use the following as an example of troubleshooting API changes for picklist attribute values in custom implementations:

In the sample response, there are several nodes, and there is also an array under the label values. Assume a certain picklist moved from a single value to multi-value, meaning that when you query the API that uses that picklist, instead of one value being returned, an array is returned with all the values selected. For example, an order with a picklist for status that contains values for draft, in progress, or activated would return all those values if the picklist becomes multi-valued. This changes the way the object is parsed by the API on the other side. Instead of receiving a string with the represented value, you receive an array with all the values. Therefore, everything that was expecting a string will fail. If you investigate each element in the array, you can see the values returned and determine the impact on the custom implementation.

Configuring CME Triggers

This section explains how to configure custom CME trigger settings.

Enabling the AllTriggers Custom Setting

To use Vlocity Communications, Media, and Energy, the AllTriggers custom setting must be set to on. These triggers generate global keys.

1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. Click **Custom Settings**.
3. Next to Trigger Setup, click **Manage**.
   If you don't see Trigger Setup, click the **more** link at the bottom of the page until it is displayed.
4. Click **AllTriggers**.
   If AllTriggers is not present, create the entry:
   a. Click **New**.
   b. On the **Trigger Setup Edit** page, enter the following information:
      - **Name** is **`AllTriggers`**
      - Select **Trigger On**.
5. Ensure the **Trigger On** checkbox is selected.
   If it is not, click **Edit** and select **Trigger On**.
6. Click **Save**.

Enabling the Party Model Trigger

The Party.EnablePartyModel trigger controls party creation and plays a role in creating communities. This trigger is not enabled by default. If you plan to use the party relationship and the relationship graph features, you must enable the Party.EnablePartyModel trigger when you install or upgrade Vlocity Communications, Media, and Energy.

Enabling the party model will increase the storage requirements in your org. In general terms, an additional party record will be generated for each account, contact, and Vlocity household record stored.

To enable the party model trigger:

1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. Click **Custom Settings**.
3. Next to Trigger Setup, click **Manage**.
4. On the **Trigger Setup** page, next to **Party.EnablePartyModel**, click Edit.
5. Ensure the **Trigger On** checkbox is selected. If it is not, click **Edit**. Select **Trigger On** and click **Save**.

Administration Jobs Reference for Vlocity Communications, Media, and Energy

There are multiple maintenance, upgrade, EPC, and other jobs for configuring Vlocity Communications, Media, and Energy. This guide describes each job in detail.

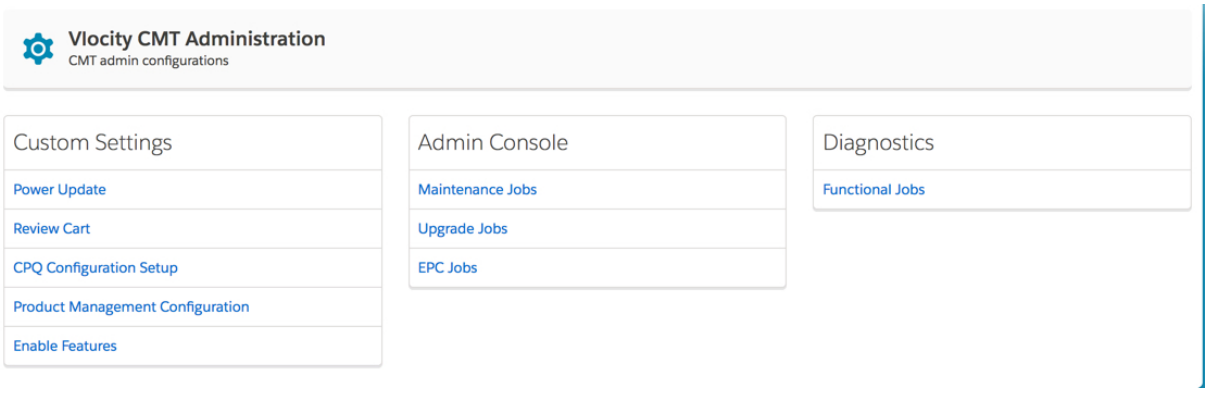
In most cases, do not run administration jobs on live production environments. Running the jobs may disrupt order processing and the customer experience.

The Vlocity CMT Administration tab includes custom settings and the Admin Console, from which you can the administration jobs.

Accessing the Vlocity CMT Administration Tab

To go to the Vlocity CMT Administration tab:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.



Running Maintenance Jobs

If you are installing or upgrading Vlocity Communications, you must run some Vlocity maintenance jobs as part of the post-installation or upgrade process. See CMT Maintenance Jobs Fresh Install vs. Upgrade.

To go to the Vlocity CMT Administration Maintenance jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Admin Console**, click **Maintenance Jobs**.
4. Next to the maintenance job that you want to run, click **Start**.

CMT Maintenance Jobs Fresh Install vs. Upgrade

This table displays when you must run a maintenance job during a fresh install or an upgrade.

To see details about each job, click the link in the **Maintenance Job Name** column.

Maintenance Job Name	Fresh Install	Upgrade to Spring '21 from any release since Winter '18	Upgrade to Winter '21 from any release since Winter '18	Upgrade to Fall '20 from any release since Winter '18	Upgrade to Spring '20 from any release since Winter '18	Upgrade to Winter '20 from Fall '19	Upgrade to Fall '19 from Summer '19
Account Hierarchy Maintenance	Yes	No	No	No	No	No	No
Interface Implementation Maintenance (Restore)	Yes	No	No	No	No	No	No
Interface Implementation Maintenance (Merge)	No	Yes	Yes	Yes	Yes	Yes	Yes
Field Maps Maintenance	Yes	Yes (Warning: will override existing data)	Yes (Warning: will override existing data)	Yes (Warning: will override existing data)	Yes (Warning: will override existing data)	Yes (Warning: will override existing data)	No (Warning: will override existing data)
Object Maps Maintenance	Yes	No	No	Yes (Warning: will override existing data)	No (Warning: will override existing data)	No (Warning: will override existing data)	No (Warning: will override existing data)
Product Hierarchy Maintenance	No	Yes	Yes	Yes	Yes	Yes	Yes
Clear Managed Platform Cache	No	Yes	Yes	Yes	Yes	Yes	Yes
Refresh Platform Cache (formerly Refresh Pricebook)	No	Yes	Yes	Yes	Yes	Yes	Yes

Account Hierarchy Maintenance

This maintenance job adds any missing root account fields for hierarchical accounts.

Applies to	All releases
Apex Name	`ResolveRootAccountBatchJob`

Where to Run This Job	CMT Administration tab
Risks of Running This Job	None
When Not to Run This Job	N/A
When to Run This Job	• After migrating hierarchical account data to ensure that the migrated data has valid root account IDs. • Anytime that you find any root account IDs are missing.
More Information	The Account Hierarchy Maintenance job resolves the root account field for all accounts in the org. It is especially useful if your org has complex account hierarchies and relationships, because during data migration, it is likely that IDs must be resolved in the target org. The **Root Account** (**vlocity_cmt__RootAccountId__c**) field stores the root account ID for each account in the hierarchy. The **Root Account** field is essential to maintain relationships between accounts. The Account Hierarchy Maintenance job finds any accounts that have a null root account field and then adds the root account ID. The job reports any errors encountered in the job status email it sends to the org administrator. After running the Account Hierarchy Maintenance job, you can run the below query to validate the data in your org: 1. Open the Developer Console. 2. In the Query Editor tab, paste the following SOQL: `SELECT COUNT() FROM Account WHERE vlocity_cmt__IsRootResolved__c = false AND ParentId != null;` 3. Click **Execute**. If the query returns 0 rows, there are no accounts that need to be updated.

You can run this query to validate that the Account Maintenance Hierarchy job completed successfully:

```
SELECT COUNT() FROM Account WHERE vlocity_cmt__IsRootResolved__c = false AND
ParentId != null;
```

The query should return zero rows.

Batch Validation

This job validates a list of header items in opportunities, orders, or quotes against a predefined list of rules and entity filters.

Applies to	All releases
Apex Name	`XLIBatchValidationService`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	HIGH RISK Contact Vlocity Support for guidance before using this job.
When Not to Run This Job	Contact Vlocity Support.
When to Run This Job	Contact Vlocity Support.

Clear Managed Platform Cache

This job clears the org cache in the CPQPartition platform cache. The job does not affect other cache partitions.

Applies to	Winter '18 and later
Apex Name	`clearPlatformCache` in `TelcoAdminConsoleHandler`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	All product hierarchy and context rules data will be deleted from the org cache, which may affect runtime performance.
When Not to Run This Job	N/A
When to Run This Job	After making changes to context rule sets, context rules or context rule conditions.
More Information	The Clear Managed Platform Cache job deletes all product hierarchy and context rules data in the org cache. You can restore product hierarchy data by running the Refresh Platform Cache job. Context rules data will be repopulated when the rules are invoked at runtime. You can use the following Apex method to programmatically run the Product Hierarchy Maintenance, Clear Managed Platform Cache, and Refresh Platform Cache jobs in sequence: `CMTAdminJobService.startProductHierarchyMaintenanceJob(pricebookId)` To run these jobs on all price books, substitute `null` for the price book ID.

Field Maps Maintenance

This job creates or restores default Field Mapper settings that the CPQ flow uses to map fields between objects. This job is critical for obtaining the latest field mappings for new fields.

You can run the Field Maps Maintenance job as part of the upgrade process when you want to restore changes to the default field mappings back to factory settings.

IMPORTANT

This job deletes any custom maps, so record any custom field mappings you need to restore after this job has run.

Applies to	All releases
Apex Name	`processFieldMapperData` in `TelcoPostInstall`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	**High risk**: Completely removes existing field mappings and restores the default mappings.
When Not to Run This Job	Do not run the Field Maps Maintenance job on any environment without proper planning. Never run the Field Maps Maintenance job on live production environments.

When to Run This Job	• After planning and documenting existing field mappings • To restore or recreate default field mappings • After upgrades that introduce new fields to map • On development or test environments, and then on production environments, but not on live production environments
More Information	If you have custom mappings to retain, record your customization details before running the Field Maps Maintenance job. You can use the Field Maps Maintenance job as a restore job. The Field Maps Maintenance job may be helpful for troubleshooting, for example, to resolve issues in Vlocity Cart. For more information, see Mapping Fields Using Field Mapper.

Interface Implementation Maintenance (Merge)

This job merges existing interfaces and implementations with the default Vlocity interfaces and interface implementations.

Applies to	All releases
Apex Name	`mergeInterfaceImplementation` in `TelcoPostInstall`
Where to Run this Job	CMT Administration tab
Risks of Running This Job	None
When Not to Run This Job	N/A
When to Run This Job	Run the Interface Implementation Maintenance (Merge) job to: • Import the latest Vlocity interfaces or implementations. • Retain existing interfaces and ensure they remain unchanged.
More Information	The Interface Implementation Maintenance (Merge) job adds any missing interfaces or implementations. For more information, see Interfaces, Implementations, and Services.

Interface Implementation Maintenance (Restore)

This Maintenance job restores the default set of Vlocity Communications interfaces and implementations, returning the system to the default operational state.

Applies to	All releases
Apex Name	`processInterfaceImplementation` in `TelcoPostInstall`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	Removes any modifications made to the default interfaces and implementations. Returns all active and default flags on interface implementations to their original states.
When Not to Run This Job	If you have custom implementations to retain, especially in production environments
When to Run This Job	Run only on a first-time package installation.
More Information	Run only on a first-time package installation.The Implementation Maintenance (Restore) removes any interface implementation customizations in your org. Record any custom implementations you make. For more information, see Interfaces, Implementations, and Services.

Object Maps Maintenance

The Object Map Maintenance job is part of the upgrade process when changes have been made to the default object mappings. This job deletes existing object mappings and creates the following default Object Mapper settings:

- Opportunity to quote
- Quote to order
- Order to asset
- Asset-related objects

The CPQ flow uses the Object Map Maintenance job to map source opportunities to destination opportunities.

Applies to	All releases
Apex Name	`processObjectMapperData` in `TelcoPostInstall`
Where to Run this Job	CMT Administration tab
Risks of Running this Job	**High Risk.** The Object Map Maintenance job completely removes existing object mappings and restores the default mappings.
When Not to Run this Job	Do not run the Object Map Maintenance job on any environment without proper planning.
When to Run this Job	• After planning and documenting existing object mappings. • To restore or re-create default object mappings • After upgrades that introduce new objects to map • On development or test environments, and then on production environments, but not on live production environments
More Information	If you have custom implementations to retain, record the customization details before running the Object Map Maintenance job. You can use the Object Map Maintenance job as a restore job. The Object Map Maintenance job may be helpful for troubleshooting, for example, to resolve issues in Vlocity Cart. For more information, see Mapping Objects for Asset-Based Ordering.

Product Hierarchy Maintenance

This job builds a streamlined version of the product hierarchies in the Data Store sObjects, enabling the Refresh Platform Cache (formerly Refresh Pricebook) job to use the hierarchy data.

Applies to	All releases
Apex Name	`ResolveProductHierarchyBatchJob`
Where to Run This Job	Maintenance Jobs in the Admin Console section on the CMT Administration tab.
Risks of Running This Job	The Product Hierarchy Maintenance job deletes all records in the Data Store sObject with a Product Hierarchy record type after it builds the new records.
When Not to Run This Job	N/A

When to Run This Job	• After any changes to the product hierarchy or product cardinality in the Vlocity Product Console • After upgrading Vlocity Communications • After spinning a new org • Before running the Refresh Platform Cache job
Delete Old Data Option	With Fall '20 and later, you can enable this option to delete the older, inactive dataset. Only the new, rebuilt dataset is available. Enabling this option can save storage space. See the More Information section about how datasets are used for this job.
More Information	The Product Hierarchy Maintenance job is a global job that reads the product child items across the shared catalog and creates a flat version of the product hierarchy in the Data Store sObject to enable quick access to child record IDs at runtime. Messages appear in the CMT Administration screen after successfully completing the Product Hierarchy Maintenance job and administrators receive email notification. After running this job, click Refresh Platform Cache to run the ProductHierarchyBatchProcessor and copy the new product hierarchy data to the platform cache. With Fall '20, running the Product Hierarchy Maintenance job no longer requires downtime. Users can continue to work while the job runs. The Product Hierarchy Maintenance job uses a new cached data set, and there are two sets of data: the current one which is active and the new one that the job is building, which is inactive. Any changes to Products and the PCIs are recorded in the new dataset. When PHM completes, the older dataset is deactivated, and the new one is activated. Vlocity Mobile also uses the Data Store sObject to store application definition records. If you are using the Vlocity Mobile application, ensure that your mobile application definition does not create records using the Product Hierarchy record type. Otherwise, the Product Hierarchy Maintenance job deletes them. You can use the following Apex method to programmatically run the Product Hierarchy Maintenance, Clear Managed Platform Cache, and Refresh Platform Cache jobs in sequence: `CMTAdminJobService.startProductHierarchyMaintenanceJob(`*`pricebookId`*`)` To run these jobs on all price books, substitute `null` for the price book ID.
Troubleshooting	If the following error occurs: `SendEmail failed. First exception on row 0; first error: NO_MASS_MAIL_PERMISSION, Single email is not enabled for your organization or profile.` Go to Setup > Deliverability. Ensure that **Access Level** is set to **All Email**.

Refresh Platform Cache

This job copies product hierarchy data to the platform cache and rebuilds the product attribute cache.

Run the Product Hierarchy Maintenance job before running Refresh Platform Cache in order for changes in product bundles to be reflected in your implementation.

NOTE

In the Fall '18 release and earlier, this job is named Refresh Pricebook.

Applies to	Winter '19 and later releases

Apex Name	`ProductHierarchyBatchProcessor`
	
	`ProductAttributesBatchProcessor`

Where to Run This Job	CMT Administration tab

Risks of Running This Job	HIGH RISK
	
	Can cause orders to fail in a live production environment.
	
	Because this job includes price books that Vlocity doesn't use, it can take longer to complete if you have more price books. The time for completing this job run can vary. You can exclude non-Vlocity price books by running the following code:
	
	1. From **Setup**, click **Developer Console**.
	2. From the **Debug** menu, click **Open Execute Anonymous Window**.
	3. Enter the appropriate code, and click **Execute**.
	4. Enter the price book ID.
	
	`Map<String, Object> input = new Map<String, Object>{'methodName' => 'refreshPriceBook', 'priceBookId' => 'price book ID'};`
	
	`vlocity_cmt.TelcoAdminConsoleController controllerClass = new vlocity_cmt.TelcoAdminConsoleController();`
	
	`controllerClass.setParameters(JSON.serialize(input));`
	
	`controllerClass.invokeMethod();`

When Not to Run This Job	In a live production environment.
When to Run This Job	After changing product bundles, product cardinality, or filterable attributes in Vlocity Product Console.
More Information	The Refresh Pricebook job runs on all products of an active price book. It doesn't filter Salesforce Industries products. The Refresh Pricebook job calls two jobs to: • Copy the product hierarchies in the data store to the org cache in the platform cache for every product bundle. • Copy filterable product attributes to the sObject cache that stores filterable attributes. If the Refresh Pricebook job can't refresh one of the price books, the job moves to the next price book. It notifies you if refreshing any of the price books fails. Starting in the Winter '18 release, this job (formerly Refresh Pricebook) is available in the Maintenance Jobs list in the CMT Administration screen. For instructions on how to use this job in earlier releases, see Refreshing the Price Book Prior to Winter '18. You can use the following Apex method to programmatically run the Product Hierarchy Maintenance, Clear Managed Platform Cache, and Refresh Platform Cache jobs in sequence: `CMTAdminJobService.startProductHierarchyMaintenanceJob(`*`pricebookId`*`)` To run these jobs on all price books, substitute `null` for the price book ID. If you see the Too many DML rows: *nnnn* error message, increase the value of the PriceBookRefreshBatchSize configuration setting to eliminate it next time you run this job. You can configure the ProductAttributesBatchProcessorSize and ProductHierarchyBatchProcessorSize configuration settings to avoid exceeding Salesforce governor limits.

Refresh Pricebook

This job copies product hierarchy data to the platform cache and rebuilds the product attribute cache.

Run the Product Hierarchy Maintenance job before running Refresh Pricebook in order for changes in product bundles to be reflected in your implementation.

NOTE

In the Winter '19 release and later, this job is named Refresh Platform Cache.

Applies to	Fall '18 and earlier releases

Apex Name	`ProductHierarchyBatchProcessor`
	
	`ProductAttributesBatchProcessor`

Where to Run This Job	CMT Administration tab
Risks of Running This Job	HIGH RISK May cause orders to fail in a live production environment.
When Not to Run This Job	In a live production environment.
When to Run This Job	• After making changes to product bundles, product cardinality, or filterable attributes in Vlocity Product Console • After running the Product Hierarchy Maintenance job
More Information	The Refresh Pricebook job calls two jobs to: • Copy the product hierarchies in the data store to the org cache in the platform cache for every product bundle. • Copy filterable product attributes to the sObject cache that stores filterable attributes. If the Refresh Pricebook job cannot refresh one of the price books, the job moves to the next price book, notifying you if refreshing any of the price books fails. Starting in the Winter '18 release, the Refresh Pricebook job is available in the Maintenance Jobs list in the CMT Administration screen. For instructions on how to use the Refresh Pricebook job in earlier releases, see Refreshing the Price Book Prior to Winter '18. You can use the following Apex method to programmatically run the Product Hierarchy Maintenance, Clear Managed Platform Cache, and Refresh Pricebook jobs in sequence: `CMTAdminJobService.startProductHierarchyMaintenanceJob(`*`pricebookId`*`)` To run these jobs on all price books, substitute `null` for the price book ID. If you see the Too many DML rows: *nnnn* error message, you can increase the value of the PriceBookRefreshBatchSize custom setting to eliminate it next time you run this job.

Reset XLI Validation Data

This job resets header items in objects to their default states.

Applies to	All releases
Apex Name	`resetXLIBatchValidtionData` in `TelcoAdminConsoleHandler`
Where to Run this Job	CMT Administration tab
Risks of Running This Job	**High Risk.** Contact Vlocity Support for guidance before using this job.
When Not to Run This Job	Contact Vlocity Support.
When to Run This Job	Contact Vlocity Support.
More Information	The Reset XLI Validation job resets header items in opportunities, orders, or quotes to their original states, including header items in validated orders.

Running Upgrade Jobs

This section describes each Vlocity Communications, Media, and Energy upgrade job.

To go to the Vlocity CMT Administration upgrade jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Admin Console**, click **Upgrade Jobs**.
4. Next to the upgrade job that you want to run, click **Start**.

Attribute Model Upgrade

This job is comprised of two smaller jobs that convert a JSON field into two smaller fields which are then populated.

The jobs must be run following the steps provided in Upgrading JSON Attributes to the CME Fall '18 Schema.

Product Conversion Upgrade Job

Applies to	Upgrades to Vlocity Communications, Media, and Energy Fall '18 release New installs of Vlocity Communications, Media, and Energy Fall '18 release
Apex Name	`ConvertProductAttributeToV2BatchJob` and `ConvertXliAttributeToV2BatchJob`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	Low Risk
When Not to Run This Job	• Do not run in a live production environment when orders are being taken. • Do not run prior to CME Fall '18. • Do not run without referencing Upgrading JSON Attributes to the CME Fall '18 Schema.
When to Run This Job	• When upgrading to Vlocity Communications, Media, and Technology Fall '18. • When installing CME Fall '18 for the first time. • After referencing the considerations Upgrading JSON Attributes to the CME Fall '18 Schema.
More Information	Analyzes the existing JSONAttribute__c content on Product2, generates the new format for the JSON and adds it to the AttributeMetadata__c and AttributeDefaultValues__c fields.

Line Item Attributes Conversion

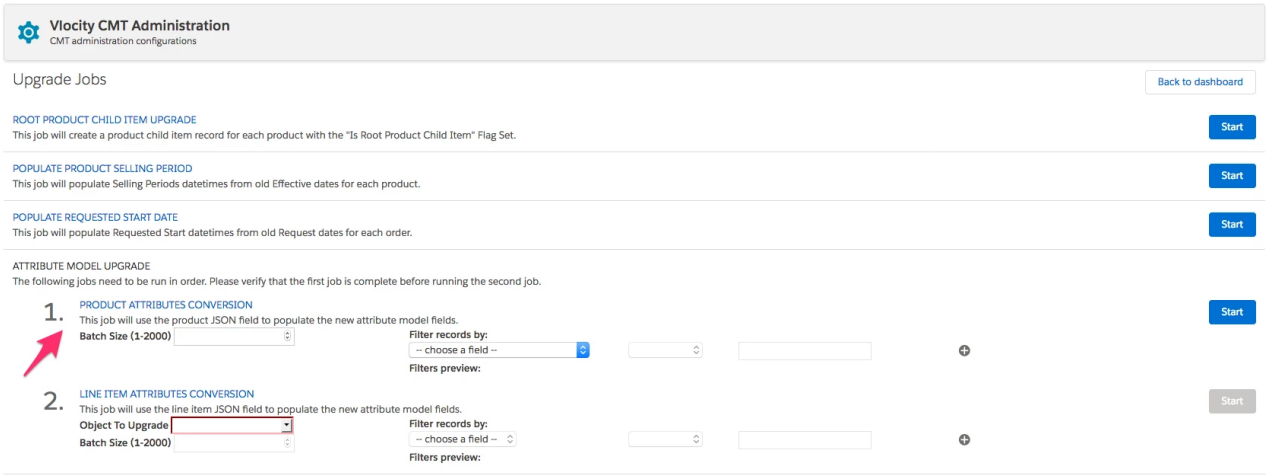
Applies to	Upgrades to Vlocity Communications, Media, and Energy Fall '18 release New installs of Vlocity Communications, Media, and Energy Fall '18 release
Apex Name	`ConvertXliAttributeToV2BatchJob`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	Low Risk
When Not to Run This Job	• Do not run in a live production environment when orders are being taken. • Do not run prior to CME Fall '18. • Do not run without referencing Upgrading JSON Attributes to the CME Fall '18 Schema.
When to Run This Job	• When upgrading to CME Fall '18. • When installing CME Fall '18 for the first time. • After referencing the considerations Upgrading JSON Attributes to the CME Fall '18 Schema.
More Information	Analyzes the JSONAttribute__c content on XLIs (Order Item - Opportunity line item - Quote line item - Asset), generates the new format for the JSON and adds it to the AttributeMetadataChanges__c and AttributeSelectedValues__c fields.

Product Attributes Conversion Job

If you enable JSONAttribute v2, you need to populate two new fields on the entity filter:

- Attribute Metadata Field Name
- Attributes Values Field Name

Run the Product Attributes Conversion job once when you enable JSONAttribute v2 to add and populates these new fields for product records.

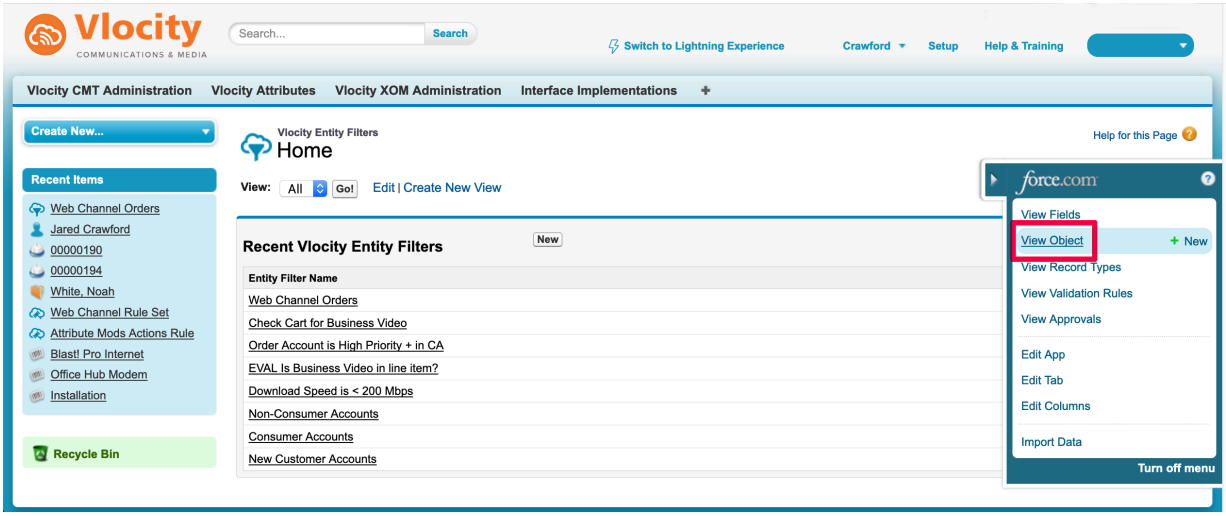


For information about running this job, see Running Upgrade Jobs and Upgrading the Attribute Schema.

If you enable JSONAttribute v2 and then create new entity filters, you must manually update these fields after adding them in the layout.

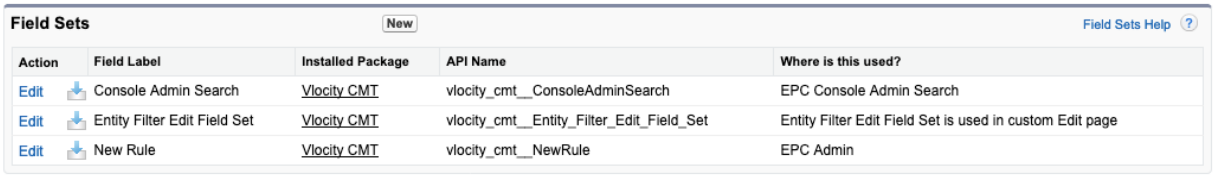
To expose these fields in the layout:

1. In Salesforce Classic experience, go to All Tabs > Vlocity Entity Filters.
2. Expand the Quick Access menu and select **View Object**.

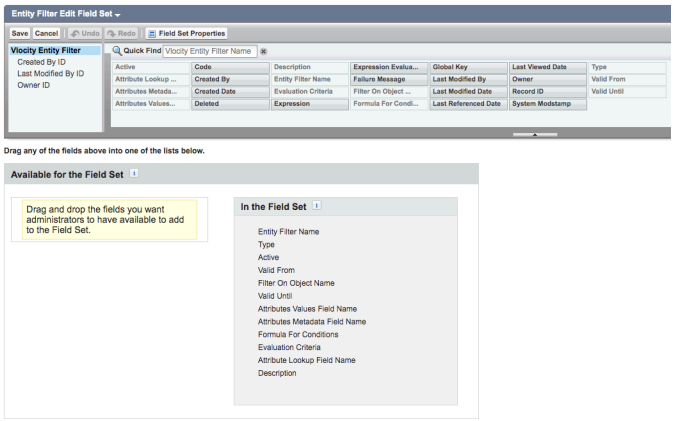


The Custom Object details appear.

3. Scroll down to the Field Sets section.

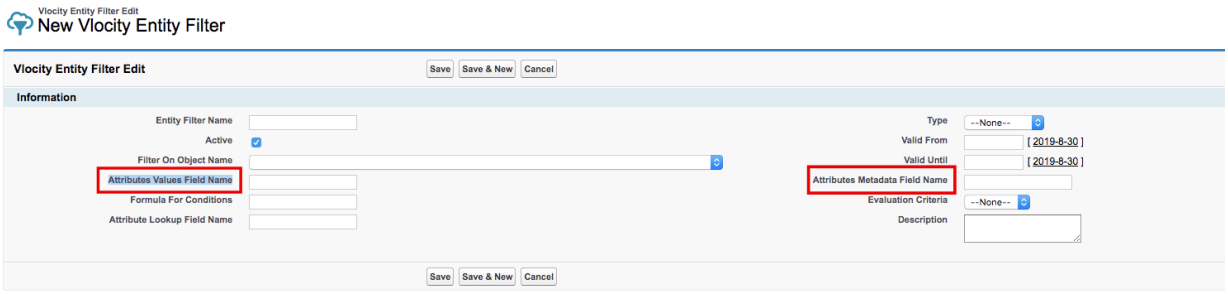


4. Next to Entity Filter Edit Field Set, click Edit.



5. Add Attribute Metadata Field Name and Attributes Values Field Name to the layout.
6. Click Save.

   The Attribute Metadata Field Name and Attributes Values Field Name fields are available in the layout when you create a New Entity Filter and then manually populate it.



Populate Product Hierarchy Global Key Path

This job populates the product hierarchy global key path from the product hierarchy path for Promotion Item and Override Definition objects.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Fall '18 release to later releases.
Apex Name	`PopulateProductHierarchyGlobalKeyPathBatchJob`
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running this Job	Low Risk
When Not to Run this Job	Do not run in a live production environment. Not needed for a fresh installation.
When to Run this Job	When upgrading to Winter '19 or later releases.

Populate Requested Start Date

This job populates Requested Start datetimes from old Request dates for each order.

Applies to	Upgrades from Vlocity Communications, Media, and Energy v15 release
Apex Name	`PopulateRequestedStartDateBatchJob`
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running This Job	**Low Risk.** This job populates the new Requested Start Date field that is the new date field for submission of an order.
When Not to Run This Job	N/A
When to Run This Job	When upgrading to v.103.x or later releases. During the upgrade, this job populates the new Requested Start Date field with values from the old Request Date field.
More Information	The FDO orders from v103 will now use the requested start date field, which is a date time field for submit orders.

Populate Missing Action Fields in XLIS

This job queries all XLIs (OrderItem, QuoteLineItem, OpportunityLineItem records) that do not have Action_c field populated, uses the ProvisioningStatus__c field value to derive value for the Action__c field and updates Action__c.

Table 4.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Fall '19 release to Summer '19 release.

Apex Name	• PopulateActionInQuoteItemJob • PopulateActionInQuoteItemJob • PopulateActionInOpptyItemJob
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running This Job	Low to no risk as this job populates the field only when the value is null.
When Not to Run This Job	This job need not be run by customers who do not use custom code to directly insert records into XLI objects. However, running the job always is low risk.
When to Run This Job	When upgrading to Summer '19 or later releases.
More Information	

Create Relationship Records

This upgrade job allows existing customers to update an existing order and it creates the required relationship records for Movement.

Table 5.

Applies To	Upgrades from Vlocity Communications, Media, and Energy Fall '19 release to Summer '19 release.
Apex Name	CreateItemRelationshipsBatchJob
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running This Job	Low Risk
When Not to Run This Job	
When to Run This Job	When upgrading to Summer '19 or later releases.
More Information	

Root Product Child Item Upgrade

This job creates a root product child item for each product for any missing root product child items.

Applies to	All Releases
Apex Name	`RootProductChildItemBatchJob`
Where to Run this Job	CMT Administration tab
Risks of Running This Job	Low Risk
When Not to Run This Job	Do not run in a live production environment when orders are being taken.
When to Run This Job	• When root product child item data has been inadvertently deleted, to restore the data to fix records. • If you find the root product child items are missing for products. • When you have installed Vlocity for the very first time.
More Information	The root product child item records enable you to specify cardinality parameters for a root product. If the root product child item is missing for any products, this job helps ensure that this critical data is present.

Upgrade Encrypted Field on Attribute Object

This job updates the Encrypted field on Attribute records based on attribute assignment data.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Fall '18 release to Winter '19 release
Apex Name	`UpdateEncryptAttributeJob`

Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running This Job	Low Risk
When Not to Run This Job	Do not run in a live production environment. Not needed for a fresh Winter '19 installation.
When to Run This Job	When upgrading to Winter '19 or later releases.
More Information	

Upgrade EPC Schema

Run these jobs to upgrade the existing product data to populate new fields that support the Product Versioning feature. The Product Versioning feature tracks versions of products, picklists, and object types. These fields must be populated and maintained regardless of whether Versioning is enabled.

IMPORTANT

Ensure the **Populate New Fields** job is completed successfully before running the **Populate Product Hierarchy Group Key Path** job.

Populate New Fields

Populates new fields on all existing record types for the listed objects.

NOTE

Before running this job, deactivate flows for custom record types to avoid interference with the update. See Deactivating a Flow and Vlocity Triggers. Active them again after the job completes.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Spring '20
Apex Name	`VersioningUpgradeBatchJob`
Where to Run this Job	**Upgrade** Jobs in the Admin Console section on the CMT Administration tab.
Risks of Running This Job	Low Risk. These fields are populated only when they are null, so if you run this job again, it does not modify existing field values. By default, this job processes all products, including those that are not in the Vlocity catalog. While there is no risk to the non-Vlocity products, processing these products adds to the time to deploy.
When Not to Run This Job	Do not run this job for product configuration that is not managed directly in Enterprise Product Catalog (EPC).

When to Run This Job	• Run this job once during the upgrade process for existing product data. • Run this job for products imported from external sources. • Re-run this job for products imported from external sources after the initial run. • If you create entries in Vlocity Product Console, run this job again before updating to Vlocity Product Designer.

More Information	This batch job is used for the following versioned objects:
	
	• Product2
	  • Populates GlobalGroupKey__c with auto-generated GUID.
	  • Sets the Version Label field if you provide it. The default value is V1.0.
	• Object Type
	  • Populates GlobalGroupKey__c with an auto-generated GUID.
	  • Sets the Version Label if you provide it. The default value is V1.0.
	• Picklist
	  • Populates GlobalGroupKey__c with an auto-generated GUID.
	  • Sets the Version Label field if you provide it. The default value is V1.0.
	
	If you do not want to update custom record types, use the following steps to run the job from the Developer Console.
	
	1. From Setup, choose **Developer Console**.
	2. From the Debug menu, choose **Open Execute Anonymous Window**.
	3. Paste the following statement into the Enter Apex Code dialog box, replacing namespace with the namespace for your org:
	
	`// initiate the batch process`
	`String parameters =`
	`'{"methodName":"startVersioningUpgradeBatchJob","objectString":"Product2","versi`
	`onLabel":"V1","lifecycleStatus":null,"versionUpgradeJobBatchSize":200,"objectFil`
	`tersListString":"RecordTypeId NOT IN (\'abcde\',\'fghij\',\'klmno\')"}';`
	`vlocity_cmt.TelcoAdminConsoleController consoleController = new`
	`vlocity_cmt.TelcoAdminConsoleController();`
	`consoleController.setParameters(parameters);`
	`consoleController.invokeMethod();`
	
	`// update the Installation Assistant step`
	`AsyncApexJob batchJob = [SELECT Id, CreatedDate FROM AsyncApexJob WHERE`
	`ApexClass.Name = 'VersioningUpgradeBatchJob' ORDER BY CreatedDate DESC LIMIT 1];`
	`List<UpgradeStep__c> steps = [SELECT Id, Name FROM UpgradeStep__c WHERE Name =`
	`'Populate New Fields for Product2' LIMIT 1];`
	`if(steps.isEmpty()) return;`
	`steps[0].LastCheckStatus__c = 'Complete';`
	`steps[0].LastExecutionDateTime__c = batchJob.CreatedDate;`
	`steps[0].LastExecutionStatus__c = 'Complete';`
	`update steps;`
	4. Select the statement.
	5. Click **Execute Highlighted**.
	6. Close Developer Console.

Populate Product Hierarchy Group Key Path

Populates product group keys and product hierarchy group key path fields for existing records.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Spring '20
Apex Name	`PopulateVlocityAttachmentProdGrpKeyJob` and `PopulateProdHierarchyGrpKeyPathJob`
Where to Run this Job	**Upgrade** Jobs in the Admin Console section on the CMT Administration tab.

Risks of Running This Job	Low Risk
When Not to Run This Job	Do not run this job for product configuration managed directly in EPC.
When to Run This Job	• Run this job once during the upgrade process for existing product data. • Run this job for products imported from external sources. • Re-run this job for products imported from external sources after the initial run. • If you create entries in Vlocity Product Console, run this job again before updating to Vlocity Product Designer.
More Information	This job populates product global group key references on related objects, mainly ProductHierarchyGroupKeyPath__c, in the following objects: • Opportunity Line Item • Quote Line Item • Order Item • Asset • Promotions Item • Override Definition • Vlocity Attachment ProductHierarchyPath__c stores the hierarchy path using Product2.Id, whereas ProductHierarchyGroupKeyPath__c stores the hierarchy path using Product2.GlobalGroupKey__c.

EPC Project Item Upgrade Job

This job populates the Root Item Id and Global Key fields of EPC Project item records if they are blank.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Winter '20 or earlier release to Summer '20 release
Apex Name	`WorkSetItemUpgradeBatchJob` This class cannot be executed outside of the package because it is not global. It can only be executed from the **Vlocity CMT Administration > Upgrade Jobs** view.
Where to Run this Job	Upgrade Jobs in the Admin Console section on the CMT Administration tab.
Risks of Running this Job	Low Risk
When Not to Run this Job	Not needed for a new installation.
When to Run this Job	When upgrading to Fall '20 or later releases.
More Information	• Root Item Id — Project items are hierarchical, and the Root Item Id is the record ID of the topmost item that represents the root of the hierarchy. Having the Root Item Id ensures the project items show up correctly in the hierarchical tree list view and allows the backend to load the project items efficiently. • Global Key — This is a globally unique identifier for the project item record. It is used to identify and match the project item across Salesforce orgs when you use IDX Workbench.

Multi-Service Upgrade

This job converts all group information (currently saved as an attachment) into QuoteMember's records. This job also changes the parent of QuoteGroup records from its group cart to the master quote.

Applies to	Upgrades from Vlocity Communications, Media, and Energy Winter '20 or earlier release to Summer '20 release

Apex Name	`MSUpgradeJob`
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running this Job	Medium Risk
When Not to Run this Job	Not needed for a new installation.
When to Run this Job	When upgrading to Fall '20 or later releases. Run after completing the package upgrade.
More Information	Avoid running this multiple times after an upgrade.

Update ChangesAllowed in OrderItems

When you upgrade an org to Winter ‘20 from a release prior to Communications, Media, and Energy Summer '18, the old order products do not support the same functionality because Salesforce defaults these fields to False. The Update ChangesAllowed in OrderItems job sets the IsChangesAllowed__c field value to true for all OrderItem records that belong to an order with a status of Draft.

Applies to	Upgrades from before Vlocity Communications, Media, and Energy Summer '18 release to Winter '20 release
Apex Name	`UpdateIsChangesAllowedJob`
Where to Run this Job	Upgrade Jobs link on the CMT Administration tab.
Risks of Running this Job	Low Risk
When Not to Run this Job	When upgrading from Summer '18 or later releases.
When to Run this Job	When upgrading from releases earlier than Summer '18 .

Running EPC Jobs

The Vlocity EPC jobs set up the Vlocity Product Console object classes, default layouts, pricing variables, pricing plans, and bindings.

Run the Vlocity EPC jobs when upgrading from one version of Vlocity Communications, Media, and Energy to another.

The Vlocity EPC jobs are available from the Vlocity CMT Administration tab.

To go to Vlocity CMT Administration EPC jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Admin Console**, click **EPC Jobs**.
4. Next to the EPC job that you want to run, click **Start**.

> **NOTICE**
> The Create Object Classes and Create Default Layouts batch jobs have been deprecated. Do not run them.

CME EPC Jobs Fresh Install vs. Upgrade

The Vlocity *EPC* jobs set up the shared catalog Product Console object classes, default layouts, pricing variables, pricing plans, and bindings. The Vlocity EPC jobs are available using the Vlocity CMT Administration tab. The following jobs are available:

Name	Description	First Install	Upgrade to Fall '20	Upgrade to Spring '20	Upgrade to Winter '20	Upgrade to Fall '19	Upgrade to Summer '19	Upgrade to Winter '19	Upgrade to any 2018 release
Generate Global Keys	Generates the missing global keys for records. You must run this job before all other EPC jobs.	No	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete Default Object Layouts	Deletes the layouts that the EPC create default layouts job created. Run this job if you have made changes to any layout and you want to restore the default layouts.	No	No	No	No	No	Yes	Yes	Yes
Install Default Vlocity Objects and Layouts	Using a DataPack, installs default Vlocity objects and layouts.	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Create Default Pricing Variables and Bindings	Creates default pricing variables and bindings.	Yes	Yes	No	No	Yes (For new Usage Pricing Variables)	Yes (For new Usage Pricing Variables)	Yes	Yes
Install Default Pricing Plan Data	Using a DataPack, installs the default data for pricing plans and steps. Available in Vlocity Communications, Media, and Energy Winter '18 and later. Run only if you use pricing plans.	Yes	No	No	No	No	Yes (Energy)	Yes	Yes

Name	Description	First Install	Upgrade to Fall '20	Upgrade to Spring '20	Upgrade to Winter '20	Upgrade to Fall '19	Upgrade to Summer '19	Upgrade to Winter '19	Upgrade to any 2018 release
Create Contextual Adjustment Data	Populates the required virtual object setup to control contextual manual adjustments. Run only if you use contextual adjustments.	No	No	No	No	No	No	No	No
Create Default Time Policy	Creates a default time policy with start policy as purchase date and end policy as end of plan duration.	Yes	No	No	No	No	No	No	No

NOTE

The following deprecated jobs also appear in the CMT Administration EPC Jobs list. Do not run them.

- EPC Create Object Classes (Deprecated): Creates a set of default Vlocity Object Class objects. To see the Object Class objects, go to the Vlocity Product Console.
- EPC Create Default Layouts (Deprecated): Creates a set of default layouts for the default Vlocity Object Class objects that the EPC create object classes job created. This job provides a default layout for viewing an object in the Vlocity Product Console. If there aren't any default layouts, going to any tab in the Product Console would result in a blank page. To see the layouts, go to the Vlocity Product Console and click Object. Click any object and then click Layout Management.

NOTE

In Vlocity Communications, Media, and Energy Summer '17, the EPC jobs are prefaced with EPC. In Vlocity Communications, Media, and Energy Winter '18, they are not. Also, in Vlocity Communications, Media, and Energy Summer '17, the first listed EPC job is EPC Create Default Vlocity Objects and Layouts, while in Vlocity Communications, Media, and Energy Winter '18, the first listed EPC job is Install Default Vlocity Objects and Layouts.

NOTE
When you are upgrading from Vlocity CMT V15 to Vlocity Communications, Media, and Energy Summer '17 or later, if Vlocity objects and layouts previously existed on your org, before running the EPC jobs, go to the Vlocity Objects and Object Types tab and the Vlocity Object Layouts tab and delete duplicate records. For more information, see Delete Custom Objects in the Salesforce Help.

The jobs you run depend on whether you are performing a new installation or upgrading Vlocity Communications, Media, and Energy. To run the EPC jobs, see:

- Running Vlocity EPC Jobs for a New Installation
- Running Vlocity EPC Jobs for an Upgrade
- EPC Jobs

Create Default Contextual Adjustment Data

This job creates default contextual adjustment data, and should only be run if the user wants to control manual adjustments using virtual context rules. Virtual context rules mean there is no physical representation of adjustment data. It is virtual.

You use virtual context rules to buy contextual discount. For example, a sales representative can apply a manual adjustment. Based on user profiles, you can restrict how much you apply a manual adjustment, for example, 5% for a customer.

To use contextual discounts, complete these steps to run this job without picklist value errors.

1. From **Setup > Object Manager**, search for **Vlocity Attribute** and click it.
2. In the **Fields & Relationships** section, click the **Value Type** field.
3. In the **Values** list, if the **Lookup** value is not in the list, check the **Inactive Values** list. If it's Inactive, click **Activate**.
   The **Lookup** value is moved to the **Values** list.
   If **Lookup** is not in the **Inactive Values** list, click **New**, add Lookup in the edit box, and click **Save**. The **Lookup** value is added to the **Values** list.
4. In **Vlocity CMT Administration > EPC Jobs**, click **Start** for the CREATE DEFAULT CONTEXTUAL ADJUSTMENT DATA job.
5. In the Vlocity Product Console under the Metadata section, click the search icon next to Object.
6. Search for AdjustmentData and click it.
7. In the AdjustmentData tab, select **Attributes**.
8. Click the Account attribute, set **Value Data Type** to **Lookup**, and click **Save**.
9. Repeat step 8 for the ANY, Opportunity, Order, and Quote attributes.
10. In the App Launcher, search for and open Vlocity Attributes.
11. In the List View menu, select **All**.

12. Search the list for Account, and click the Account attribute (which has **Attribute Category** set to System Attributes).
13. In the **Value Type** field, select **Lookup**, and click **Save**.
14. Click the browser back button, and repeat steps 12 and 13 for the following attributes that also have **Attribute Category** set to System Attributes: ANY, Opportunity, Order, and Quote.

See also Qualification Rules for Pricing Adjustments.

Applies to	All releases
Apex Name	`AdjustmentData`
Where to Run This Job	From the Vlocity CMT Administration page: 1. Click **EPC Jobs**. The EPC Jobs page appears. 2. Click the **Start** button beside **CREATE DEFAULTCONTEXTUAL ADJUSTMENT DATA**.
Risks of Running This Job	**Medium Risk**
When Not to Run This Job	While the network is busy, during office hours.
When to Run This Job	This is a synchronous job and should not run more than five minutes, but it is recommended to run this job after office hours. This job is not dependent on any other Admin jobs.
More Information	• Do not modify or delete the `AdjustmentData` object. • Do not modify the attributes or the object types which are linked to the System Attributes attribute category. • The Successfully Completed message appears after this job runs. • A user can see confirmation that the job has run by checking the `AdjustmentData` object in the EPCConsole Object tab.

Delete Default Objects and Layouts

This job deletes the layouts that the Install Default Vlocity Objects and Layouts job creates.

Applies to	Summer '17 and later
Apex Name	`deleteDefaultLayouts` in `EPCPostInstallService`
Where to Run this Job	From the Vlocity CMT Administration page: 1. Click **EPC Jobs**. The EPC Jobs page appears. 2. Click the **Start** button beside **CREATE DEFAULTCONTEXTUAL ADJUSTMENT DATA**. You can also run this job from the Developer Console.
Risks of Running This Job	This job deletes all default layouts and facets in Vlocity Product Console. Before running, record any configuration changes so you can re-implement them, if needed.
When Not to Run This Job	N/A
When to Run This Job	• If your layouts include any customizations and you want to restore the default layouts or facets • When you are prepared to re-create all customized layouts and facets

More Information	The Delete Default Objects and Layouts job removes **ObjectFacet__c** and **ObjectLayout__c** data. The Delete Default Objects and Layouts job may be useful if a problem occurs in a layout or facet. After running the Delete Default Objects and Layouts job, run the Install Default Vlocity Objects and Layouts job.

Create Default Pricing Variables and Bindings

This job creates default pricing variables and bindings in a fresh installation or an upgraded org.

Applies to	Summer '17 and later releases
Apex Name	`createDefaultPricingVariablesAndBindings` in `EPCPostInstallService`
Where to Run this Job	CMT Administration tab
Risks of Running This Job	This job may disrupt order capture in a live environment.
When Not to Run This Job	Do not run the Create Default Pricing Variables and Bindings job In a live production environment when orders are being taken.
When to Run This Job	• After installing Vlocity Communications, Media, and Energy for the first time. • After upgrading Vlocity Communications, Media, and Energy. • If pricing variables are modified and you encounter a problem. • To restore default pricing variables and bindings.
More Information	In Vlocity Communications, Media, and Energy Winter '18, when loyalty pricing is enabled, this job updates the loyalty pricing variable. If loyalty pricing is disabled, this job does not update the loyalty pricing variable.

Create Default Time Policy

This job will create a default time policy with start policy as purchase date and end policy as end of plan duration.

Applies to	CME Summer '19 and later releases
Apex Name	EPCPostInstallService
When to Run This Job	When doing a fresh install or upgrading to CME Winter '19 (CMT 104)
Where to Run This Job	From the Vlocity CMT Administration page: 1. Click **EPC Jobs**. The EPC Jobs page appears. 2. Click **Start** next to **CREATE DEFAULT TIME POLICY**.
Risks of Running This Job	No risk
When Not to Run This Job	You only need to run this job once. There is no risk in running it again. The system detects if the default time policy was already generated.
More Information	To use contextual discounts feature, you must run this job. This is required to define recurring adjustments for contextual discounts. This job creates a Time Policy and the name of the policy is hardcoded. This Time Policy is used as the default when creating a recurring discount. Once the name of the Time Policy is modified, recurring discounts cannot be defined.

Generate Global Keys

This job creates any missing global keys and assigns them to the correct records.

Applies to	Summer '17 and later
Apex Name	`GenerateMatchingGlobalKeyBatch`
Where to Run this Job	CMT Administration tab
Risks of Running This Job	None
When Not to Run This Job	N/A
When to Run This Job	• If you find any global keys missing. • After upgrading Vlocity Communications, Media, and Energy, but before running any other job, to generate any missing global keys for records.
More Information	DataPacks use global keys to transfer data. The global keys field value provides a reference for transferring data from one org to another, for example, pricing elements. For example, if you have a pricing element in Org A and want to migrate the element to Org B, assign a global key to each pricing element record.

Install Default Pricing Plan Data

This job installs the Vlocity default Pricing Plan DataPack.

Applies to	Winter '18 and later
Apex Name	None. Installs the VlocityDeploy Pricing Plan DataPack.
Where to Run this Job	CMT Administration tab
Risks of Running This Job	None
When Not to Run This Job	N/A
When to Run This Job	• When you are ready to set up pricing plans. • Typically, run once to obtain the default pricing plan data for the first time. After you install the default, you can customize the default plan for your deployment.
More Information	The Install Default Pricing Plan Data job does not overwrite all pricing plans, only the default pricing plan. To restore the default pricing plan, delete your entire plan and then re-install the plan using this DataPack.

Install Default Vlocity Objects and Layouts

This job installs or restores the default Vlocity objects and layout for object classes, including:

- Object class
- Object layout
- UI facet
- UI section

Applies to	Summer '17 and later releases
Apex Name	None. Installs the EpcDefaultObjects DataPack.
Where to Run this Job	CMT Administration tab
Risks of Running This Job	This job overwrites any changes made to default Vlocity objects, such as layout customizations in Vlocity Product Console.
When Not to Run This Job	N/A
When to Run This Job	• After installing Vlocity Communications, Media, and Energy for the first time. • To restore the default objects and layouts.

More Information	This job provides all object-related hierarchy elements. It imports the Vlocity Product Console default view. It also replaces the two, separate deprecated jobs for creating default object classes and layouts.

Resolving Import Errors

The Install Default Vlocity Objects and Layouts job installs or restores the default Vlocity objects and layout for object classes. When running the Install Default Objects and Layouts EPC job, if you encounter the error `Import Error: No Configuration Found: Object Class Migration and you are using an org that uses deployed code`, perform these steps.

1. Go to the Developer Console.
2. From the Debug menu, choose **Open Execute Anonymous Window**.
   The **Enter Apex Code** window opens.
3. Paste the following Apex statement into the **Enter Apex Code** window:
   `CorePostInstallClass.runDev2ProdInserts();`
4. Select the statement.
5. Click **Execute Highlighted**.
6. Exit the Developer Console.
7. Run the "EPC Create Default Objects and Layouts" or "Install Default Objects and Layouts" job again.

Running Multilanguage Support Jobs

Multilanguage support for Vlocity Product Catalog enables CPQ to render information in various languages.

To go to Vlocity CMT Multilanguage Support jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Admin Console**, click **Multilanguage Support Jobs**.
4. Next to the Multilanguage Support job that you want to run, click **Start**.

Create Translation Job

This job deletes the layouts that the Install Default Vlocity Objects and Layouts job creates.

Applies to	Summer '18 and later.
Apex Name	StringTranslationBatch
Where to Run this Job	From the Vlocity CMT Administration page: 1. Click **Enable Features**. 2. Ensure **Multilanguage Catalog Support** is enabled. 3. Ensure you have the correct configuration for objects and fields as described in Multilanguage Field Setup. 4. Click **Multilanguage Support Jobs**. 5. Click the **Start** button beside **CREATE TRANSLATION JOB**.
Risks of Running This Job	None
When Not to Run This Job	None
When to Run This Job	This job must be run by the product administrator whenever there is a change to the products.

More Information	This job provides the following data: • Job Name • Status • Total Batches • Batches Processed • Created Date • Completion Date • Status Detail

Create Cache Translation Data

This job creates cache data for products and promotions to support search and sort in `getProductList` and `getPromotionList`. This job copies data from `StringTranslation__c` and Product/Promotion objects and fills `CachedProduct2Translation__c` and `CachedPromotionTranslation__c`

Applies to	Summer '18 and later
Apex Name	CacheTranslationCatalogBatch
Where to Run this Job	From the Vlocity CMT Administration page: 1. Click **Multilanguage Support Jobs**. 2. Click the **Start** button beside **CREATE CACHE TRANSLATION DATA**.
Risks of Running This Job	None
When Not to Run This Job	None
When to Run This Job	This job must be run whenever you have created new translations.
More Information	This job provides the following data: • Job Name • Status • Total Batches • Batches Processed • Created Date • Completion Date • Status Detail

Load Default Fields Configurations

This job loads out of box object and field configurations for multilanguage support. However, this job does not remove any data you added.

Applies to	Summer '18 and later
Where to Run this Job	From the Vlocity CMT Administration page: 1. Click **Multilanguage Support Jobs**. 2. Click the **Start** button beside **LOAD DEFAULT FIELDS CONFIGURATIONS**.
Risks of Running This Job	None
When Not to Run This Job	None

When to Run This Job	This job must be run: • They very first time you are deploying translation. • Whenever there is a change to Multilanguage Field Setup. • To restore any fields you have accidently deleted.

Running Cacheable API Jobs

This section describes each Vlocity Communications, Media, and Energy Cacheable API job.

To go to the Vlocity CMT Administration EPC jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Admin Console**, click **Cacheable API Jobs**.
4. Next to the Cacheable API job that you want to run, click **Start**.

Load API Metadata

This populates the Vlocity API Metadata table.

Run the Load API MetaData job before running the Populate API Cache job.

Applies to	All releases
Apex Name	`loadAPIMetadataCAJob` in `TelcoAdminConsoleHandler`
Where to Run This Job	Cacheable API Jobs link on the CMT Administration tab.
Risks of Running This Job	**Low Risk**
When Not to Run This Job	N/A
When to Run This Job	When you need to load or refresh API metadata.
More Information	This job refreshes the records on the VlocityAPIMetadata object. You see the message Successfully Completed! after running this job.

Populate API Cache

This job processes existing data and populates the API cache.

Run the Load API Metadata job to populate the Vlocity API Metadata table before running the Populate API Cache CMT Administration job.

Applies to	All releases By default, the Populate API Cache jobs are run for all catalogs.
Apex Name	`populateCacheCAJob` in `TelcoAdminConsoleHandler`
Where to Run This Job	Cacheable API Jobs link on the CMT Administration tab.
Risks of Running This Job	The cache will be cleared and repopulated, but you will not lose any data. There will be a temporary performance impact in a production environment. Run this job only during a maintenance window.

When Not to Run This Job	Do not run this job if cacheable definitions are not ready to be cached. Only run this job once you have completed your product catalog configuration.
When to Run This Job	Cached information is a snapshot of current cacheable definitions at the time caching was invoked. Run this job when definitions are ready or when testing the cached information is required. After running this job, you will see the status, creation and completion dates of the individual API cache jobs. When you initially run this job, select the checkboxes for all cache job options because some of the jobs have dependencies on other jobs. You can select all options by selecting the **Name** checkbox on the option list after clicking **Start**.
More Information	• This job calls cache handlers that cache the catalog and product information, such as product structure, cardinality, rules, attributes, and pricing. • All cache handlers must be selected and cached at least once. • The dependency order of handlers is descending. • Once cache entries have been generated, each handler can be selectively re-executed based on which information needs to be recached. For example, the pricing cache handler may be invoked often due to frequent promotional pricing changes. • You must clear the entire contents of the cache response object if you make changes to the product catalog data.

Delete Pseudo Records

This job cleans the pseudo records (accounts and orders) created and used internally by the Digital Commerce APIs.

Available in	Winter '20 and later.
Apex Name	DELETE PSEUDO RECORDS
Where to Run This Job	Cacheable API Jobs link on the CMT Administration tab.
Risks of Running This Job	There are no risks to running this job even if accounts and orders are removed.
When to Run This Job	When you need to delete all existing pseudo records created before a specified time. You can also run this job at any time.
More Information	As part of the Vlocity Cacheable API solution, when the Create Basket API is called for an anonymous user, a pseudo-Salesforce Account record is created with random Account Name (for example: OuL48cVU4r) with a Billing Record Type. The creation of these accounts may cause maintenance issues for clients who create and run reports against the Salesforce Account object records and therefore needing to identify the Accounts we create as part of our Cacheable API solution. Starting with the CME Winter '20 release, pseudo-accounts and pseudo-orders are deleted automatically. To delete pseudo records created prior to the CME Winter '20 release, you can run the Delete Pseudo Records cacheable API job. Note that the Delete Pseudo Records job would only need to be run one time.

Delete Expired API Cache

This will clear expired cached API records created earlier than today.

Applies to	All releases
Apex Name	`startDeleteExpiredCacheJobs` method in `TelcoAdminConsoleHandler`
Where to Run This Job	Cacheable API Jobs link on the CMT Administration tab.
Risks of Running This Job	**Low risk**. This job deletes cached information that is no longer accessible due to its expiration time.
When Not to Run This Job	N/A

When to Run This Job	Run this job regularly to keep the number of cached records at a minimum.
More Information	Cacheable API information is cached in the `CachedAPIResponse` object with expiration times; the expiry date is set when you run the Populate API Cache job. Records with expiration times in the past are no longer accessible by the Digital Commerce APIs. This job deletes records that are older than midnight of the day the batch job was executed.

Running Report Jobs

To design commercial and technical catalog products that comply with TM Forum standards, you should configure offers and specifications according to the Offer Spec Realization feature introduced in CME Winter ’20. This feature contains product configurations to ensure catalog entities are correctly structured.

To run Offer Spec Maintenance report jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under Admin Console, click **Report Jobs**.
4. Next to the Report job that you want to run, click **Start**.

Audit Product Structure

To determine whether your offers are properly configured and comply with TM Forum standards, you can run the Audit Product Structure job. This job produces a report that lists products containing embedded products with mismatched Specification Type values and emails the report to your account.

Applies to	Vlocity Communications, Media, and Energy Winter ’20 release and later.
Apex Name	`ReportMismatchedSpecTypeBatchJob`
Where to Run This Job	Upgrade Jobs link on the CMT Administration Vlocity CMT Administration page → Admin Console table → Report Jobs link → Audit Product Structure.
Risks of Running This Job	Low Risk
When Not to Run This Job	Do not run as part of a fresh install.
When to Run This Job	Anytime, and when upgrading from CME releases prior to Winter ‘20.
More Information	An email is sent even when no entries are found in the content tables of the report.

Audit Product Specification Type

To determine whether your offers are properly configured and comply with TM Forum standards, you can run the Audit Product Specification Type job. This job produces a report that lists products with missing Specification Type values and emails the report to your account.

Applies to	Vlocity Communications, Media, and Energy Winter ’20 release and later.
Apex Name	`ReportNullSpecTypeBatchJob`
Where to Run This Job	Vlocity CMT Administration page → Admin Console table → Report Jobs link → Audit Product Specification Type
Risks of Running This Job	Low Risk
When Not to Run This Job	Do not run as part of a fresh install.

When to Run This Job	Anytime, and when upgrading from CME releases prior to Winter ‘20.
More Information	An email is sent even when no entries are found in the content tables of the report.

Running Functional Jobs

This section describes each Vlocity Communications, Media, and Energy Functional job.

To go to the Vlocity CMT Administration Functional jobs:

1. Click the **All Tabs** button, or click the App Launcher and then click the **Vlocity CME Admin** app.
2. Click **Vlocity CMT Administration**.
3. Under **Diagnostics**, click **Functional Jobs**.
4. Next to the functional job that you want to run, click **Start**.

For more information about the Custom Settings listed on the Functional Jobs page, see CPQ Configuration Settings Reference.

Basic Configurations

This job ensures that your org has all appropriate configuration settings and values. It checks whether any settings are missing.

Applies to	All releases
Apex Name	`CMTAdminDiagnosticService`
Where to Run This Job	CMT Administration tab
Risks of Running This Job	None
When Not to Run This Job	N/A
When to Run This Job	• After installing Vlocity Communications, Media, and Energy for the first time. • After system upgrade. • Anytime you want to determine whether your org has all the appropriate configuration settings and values.
More Information	If a setting is missing or its value is not appropriate, this job highlights the setting and the typical values.

Other Jobs

This section describes other jobs that you can run using anonymous Apex code in the Developer Console.

You can use the EPC batch jobs to fix JSONAttributes and simple rule conditions. If you created the products and assigned attributes within the Vlocity Product Console, use EPCProductAttribJSONBatchJob. If you created the products and assigned attributes within the Aloha Products page, use FixProductAttribJSONBatchJob.

EPC Product Attribute JSON

This job regenerates the JSON Attribute fields for products managed through Vlocity Product Console. For products that are managed in Product Console, you can run this batch job in case the JSONAttribute gets out-of-sync with the attributes that are assigned to the product.

First get a list of Product2 Ids, `prodIds`, then call the batch job and execute it anonymously:

```
Map<Id,Product2> products = new Map<Id<Product2>([SELECT Id FROM Product2]);
Database.executeBatch(new EPCProductAttribJSONBatchJob (products.keySet()), 1);
```

Applies to	Summer '17 and later
Apex Name	`EPCProductAttribJSONBatchJob`
Where to Run this Job	Developer Console
Risks of Running This Job	None
When Not to Run This Job	Do not run this job In a live production environment when orders are being taken. Do not run this job in orgs with products created using the Salesforce Classic interface instead of Vlocity Product Console.
When to Run This Job	• After spinning a new org. • After importing data containing product attributes.
More Information	The `EPCProductAttribJSONBatchJob` processes products attributes created with Vlocity Product Console and corrects the product attribute record IDs. It does not affect the org cache. It replaces the `FixProductAttribJSONBatchJob` used prior to the Vlocity Communications, Media, and Energy Summer '17 release. For more information, see Fix Product Attribute JSON.

EPC Fix Compiled Attribute Override

This job corrects attribute record IDs after data migration when product attribute overrides are present on promotions. This job regenerates the JSONAttribute field on CompiledAttributeOverride records for attribute overrides (for products that have attribute overrides, this job regenerates all the JSONAttributes for those overrides).

Run the batch job in **execute anonymous**:

```
Database.executeBatch(new EPCFixCompiledAttributeOverrideBatchJob (), 1);
```

If you run this job on an installed package, as is usually the case at a customer site, then run:

```
Database.executeBatch(new vlocity_cmt.EPCFixCompiledAttributeOverrideBatchJob
(), 1);
```

Applies to	CME Summer '17 and later
Apex Name	EPCFixCompiledAttributeOverrideBatchJob
Where to Run This Job	Developer Console
Risks of Running This Job	None
When Not to Run This Job	Do not run the `EPCFixCompiledAttributeOverrideBatchJob` in a live production environment when orders are being taken.
When to Run This Job	After spinning a new org and migrating data, specifically, promotions data, from an old org to a new one.
More Information	For more information, see EPC Product Attribute JSON or Fix Product Attribute JSON.

Fix Product Attribute JSON

This job Regenerates the JSONAttribute fields for products managed through the Salesforce Classic interface, instead of Vlocity Product Console.

For products that are managed in the legacy Salesforce Aloha page through the Attributes Setup component, you can run this batch job in case the JSONAttribute gets out-of-sync with the attributes that are assigned to the product. By default, this job only creates the JSONAttribute for the products where JSONAttribute is null.

To update all products:

1. Go to Vlocity CMT Administration > CPQ Configuration Setup .
2. Add a new setting:
   - Name: **ProductAttribJSONBatchJob Update All**
   - Value: **True**
3. Call the batch job in execute anonymous:
   ```
   Id batchJobId = Database.executeBatch(new FixProductAttribJSONBatchJob());
   ```

Applies to	Releases prior to Summer '17
Apex Name	`FixProductAttribJSONBatchJob`
Where to Run this Job	Developer Console
Risks of Running This Job	None
When Not to Run This Job	Do not run this job In a live production environment when orders are being taken. Do not run this job In orgs with products created using Vlocity Product Console.
When to Run This Job	• After spinning a new org. • After importing data containing product attributes.
More Information	This job processes product attributes created using the Salesforce Classic interface and corrects the product attribute record IDs. It does not affect the org cache.

EPC Create Context Dimension Batch Job

This job sets the Lookup to Context Dimension

A lookup to ContextDimension__c has been added to the EntityFilterCondition__c table as of release CME Winter '18. In earlier releases, the FieldName__c field was used to set the ContextDimension. At admin time, for a simple rule condition, it is easier and less error-prone if the Context Dimension is a lookup and not a text field. This batch job is created to populate this lookup for records created prior to CME Winter '18 release. Records created in CME Winter '18 release and later have this lookup populated.

Run the batch job in execute anonymous:

```
Database.executeBatch(new EPCCreateContextDimensionBatchJob(), 1);
```

If you run this job on an installed package, as is usually the case at a customer site, then run:

```
Database.executeBatch(new vlocity_cmt.EPCFixCompiledAttributeOverrideBatchJob
(), 1);
```

EPC Update Attribute Configurable Batch Job

This job sets the IsConfigurable__c for products with attributes created prior to the use of the Product Console. For products that were created in legacy Salesforce Aloha page through the Attributes Setup component, the IsConfigurable__c flag was used differently than it is in Product Console.

To update the flag for usage in Product Console, change the flag according to the logic:

1. If IsReadOnly__c is False and IsConfigurable__c is False, then set IsConfigurable__c to True. Otherwise, do not change IsConfigurable__c.
2. Enter a Datetime where all attribute assignments created before that date will be updated
3. Run the batch job in execute anonymous:

```
Datetime createdBeforeDate = Datetime.newInstanceGmt(2016, 1, 1, 0, 0,
0); //example, change to reflect your date
UpdateAttributeConfigurableBatchJob epcBatch = new
UpdateAttributeConfigurableBatchJob(createdBeforeDate);
Database.executeBatch(epcBatch);
```

Asset-Based Ordering Administration

Standard ordering systems end the customer relationship after the order is complete. However, not all orders are about buying new products or services. An existing customer may want to modify the existing assets or services.

Using asset-based ordering (ABO), you can manage customers' products and services throughout their life cycles.

To use asset-based ordering, you must complete the following administrative tasks:

1. Select the asset-based ordering implementation.
2. Map fields for asset-based ordering.
3. Map objects for asset-based ordering.

Applying Vlocity Page Layouts

Vlocity enhances the page layout for the following Salesforce objects:

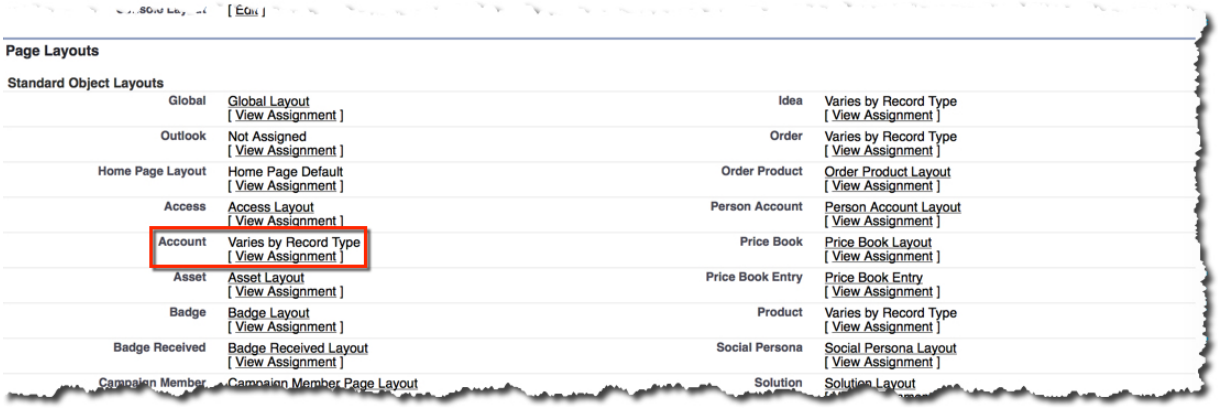
- Account (Desktop and Console)
- Asset (Console only)
- Case (Desktop and Console)
- Contact (Desktop and Console)
- Lead
- Opportunity (Desktop and Console)
- Opportunity Product
- Order
- Order Product
- Price Book
- Product
- Quote (Desktop and Console)
- Quote Line Item

For example, with Vlocity page layouts for accounts, you can see billing information reports as well as a relationship graph that illustrates that account's key connections to people and businesses.

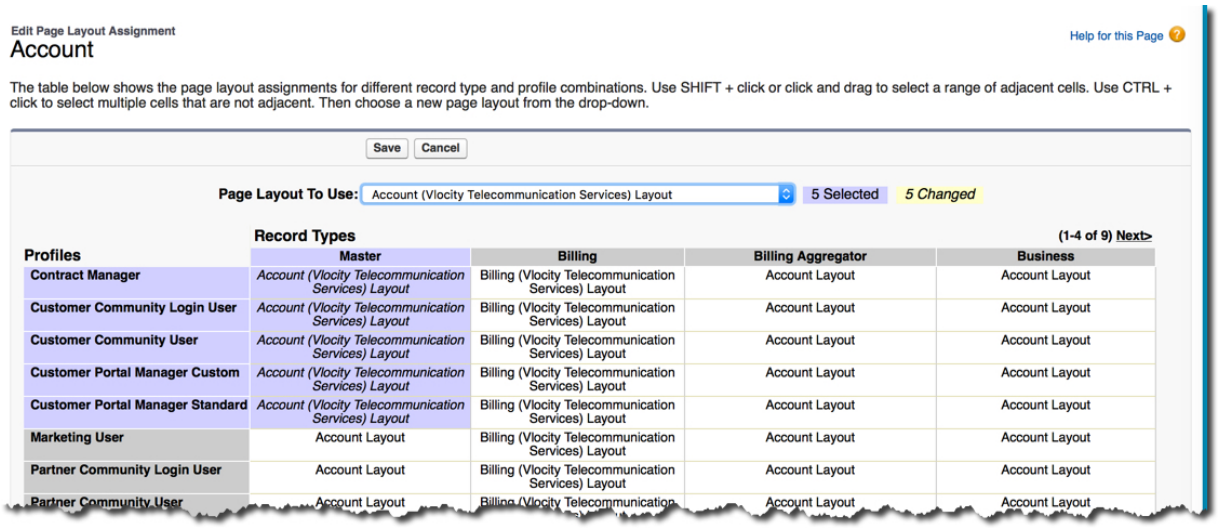
For users to see these improved page layouts and layouts for Vlocity custom objects, you must assign Vlocity layouts to user profiles.

To assign a page layout for a profile using the traditional profile user interface:

1. From Setup, in the **Quick Find** box, enter **`Profiles`**.
2. Click **Profiles**.
3. Click the profile to adjust.
4. Scroll to the **Page Layouts** section.
5. Next to Account, click **View Assignment.**



6. Click **Edit Assignment**.
7. Select the profiles that should use the enhanced Vlocity layout for Master. Press **Ctrl** to select multiple profiles.
8. In the **Page Layout To Use** picklist, select **Account (Vlocity Telecommunication Services) Layout**. If you are assigning layouts to a console user, select **Console Account (Vlocity Telecommunication Services) Layout**.


Edit Page Layout Assignment

Account

Help for this Page

The table below shows the page layout assignments for different record type and profile combinations. Use SHIFT + click or click and drag to select a range of adjacent cells. Use CTRL + click to select multiple cells that are not adjacent. Then choose a new page layout from the drop-down.

Save Cancel

Page Layout To Use: Account (Vlocity Telecommunication Services) Layout 5 Selected *5 Changed*

Record Types (1-4 of 9) Next>

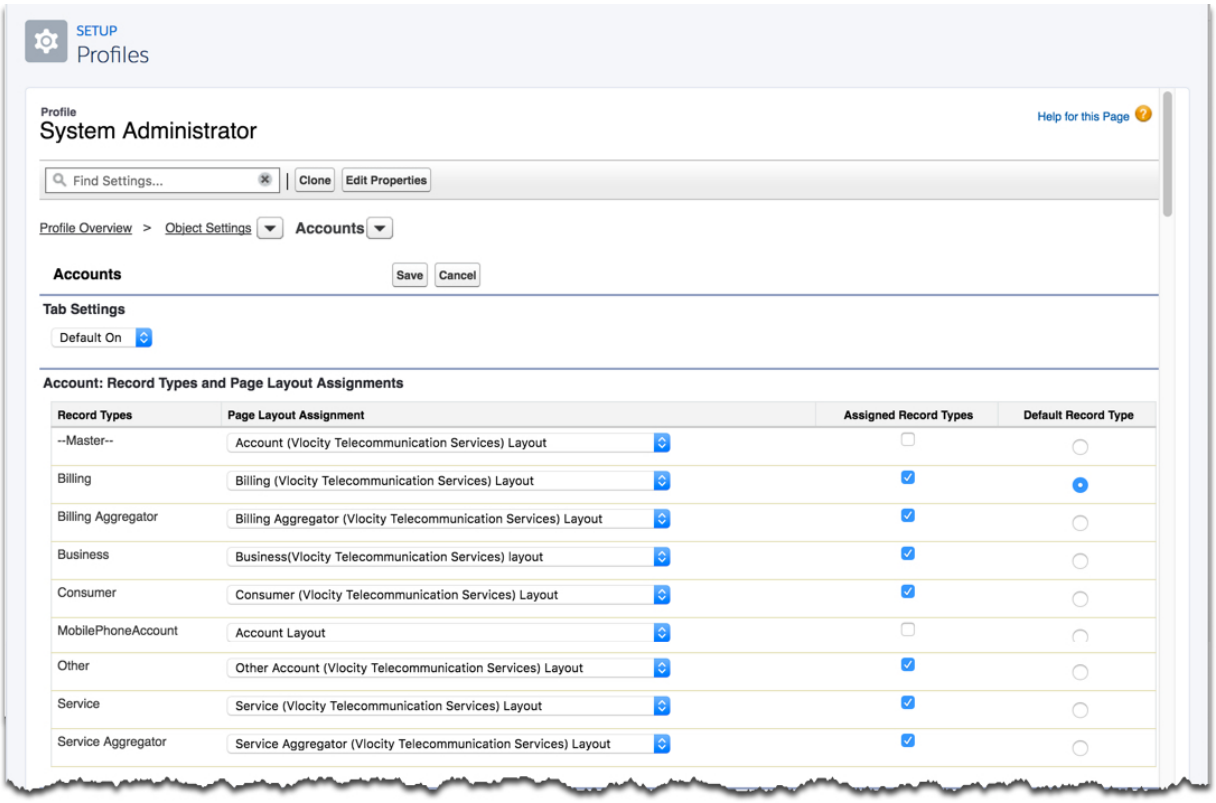
Profiles	Master	Billing	Billing Aggregator	Business
Contract Manager	*Account (Vlocity Telecommunication Services) Layout*	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Customer Community Login User	*Account (Vlocity Telecommunication Services) Layout*	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Customer Community User	*Account (Vlocity Telecommunication Services) Layout*	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Customer Portal Manager Custom	*Account (Vlocity Telecommunication Services) Layout*	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Customer Portal Manager Standard	*Account (Vlocity Telecommunication Services) Layout*	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Marketing User	Account Layout	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Partner Community Login User	Account Layout	Billing (Vlocity Telecommunication Services) Layout	Account Layout	Account Layout
Partner Community User	Account Layout	Billing (Vlocity Telecommunication	Account Layout	Account Layout

9. Select Vlocity page layouts for all other available Account record types.
   You can only view three record types at a time. If necessary, click **Next** to view and apply layouts to additional record types.
   - Billing
   - Billing Aggregator
   - Business
   - Consumer
   - Service
   - Service Aggregator
10. Click **Save**.
11. Repeat steps 2 through 9 for the following:
    - Asset (Console only)
    - Case
    - Contact
    - Opportunity
    - Opportunity Product
    - Order
    - Order Product
    - Payment Adjustment (Console only)
    - Price Book
    - Product
    - Quote

- Quote Line Item
- Household

To apply page layouts using the enhanced profile user interface:

1. From Setup, in the **Quick Find** box, enter **`Profiles`**.
2. Click **Profiles**.
3. Click the profile to adjust.
4. Click **Object Settings**.
5. Click the object for which to apply the page layout, for example, **Accounts**.
6. Click **Edit**.



7. In the Record Types and Page Layout Assignments section, for each record type, select the appropriate page layouts, for example **Account (Vlocity Telecommunications Services) Layout**. Apply page layouts for the following records:
   - Billing: **Billing (Vlocity Telecommunication Services) Layout**
   - Billing Aggregator: **Billing Aggregator (Vlocity Telecommunication Services) Layout**
   - Business: **Business (Vlocity Telecommunication Services) Layout**
   - Consumer: **Consumer (Vlocity Telecommunication Services) Layout**
   - Other: **Other Account (Vlocity Telecommunication Services) Layout**
   - Service: **Service (Vlocity Telecommunication Services) Layout**

- Service Aggregator: **Service Aggregator (Vlocity Telecommunication Services) Layout**

8. Click **Save**.
9. Repeat steps 4 through 7 for the following:
   - Asset (Console only)
   - Case
   - Contact
   - Opportunity
   - Opportunity Product
   - Order
   - Order Product
   - Payment Adjustment (Console only)
   - Price Book
   - Product
   - Quote
   - Quote Line Item
   - Household

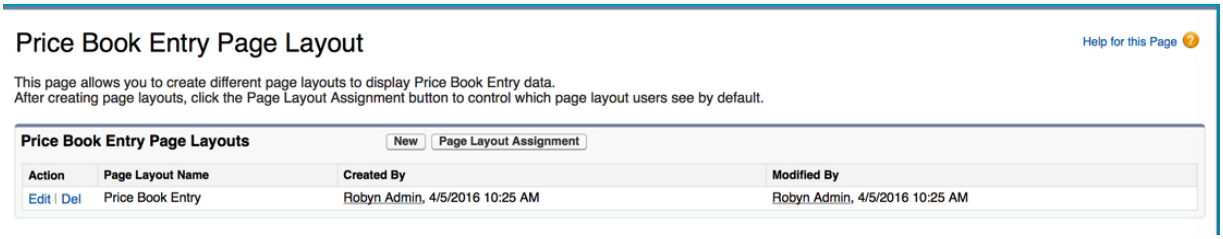
Create a New Page Layout

Some Vlocity page layouts are not packaged. You must manually create page layouts to access the Vlocity-specific fields that are associated with some objects. The steps are different for Salesforce Classic and Lightning Experience.

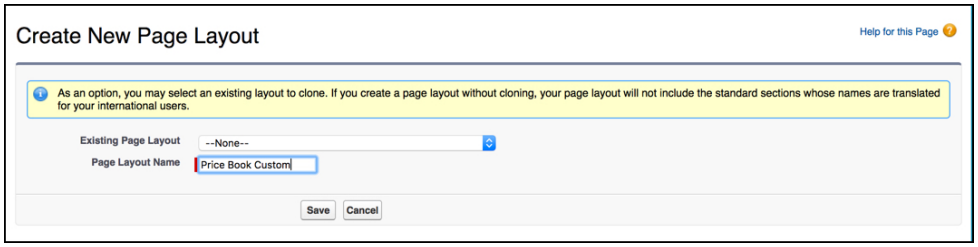
Create a New Page Layout in Salesforce Classic

To create a new page layout in Salesforce Classic:

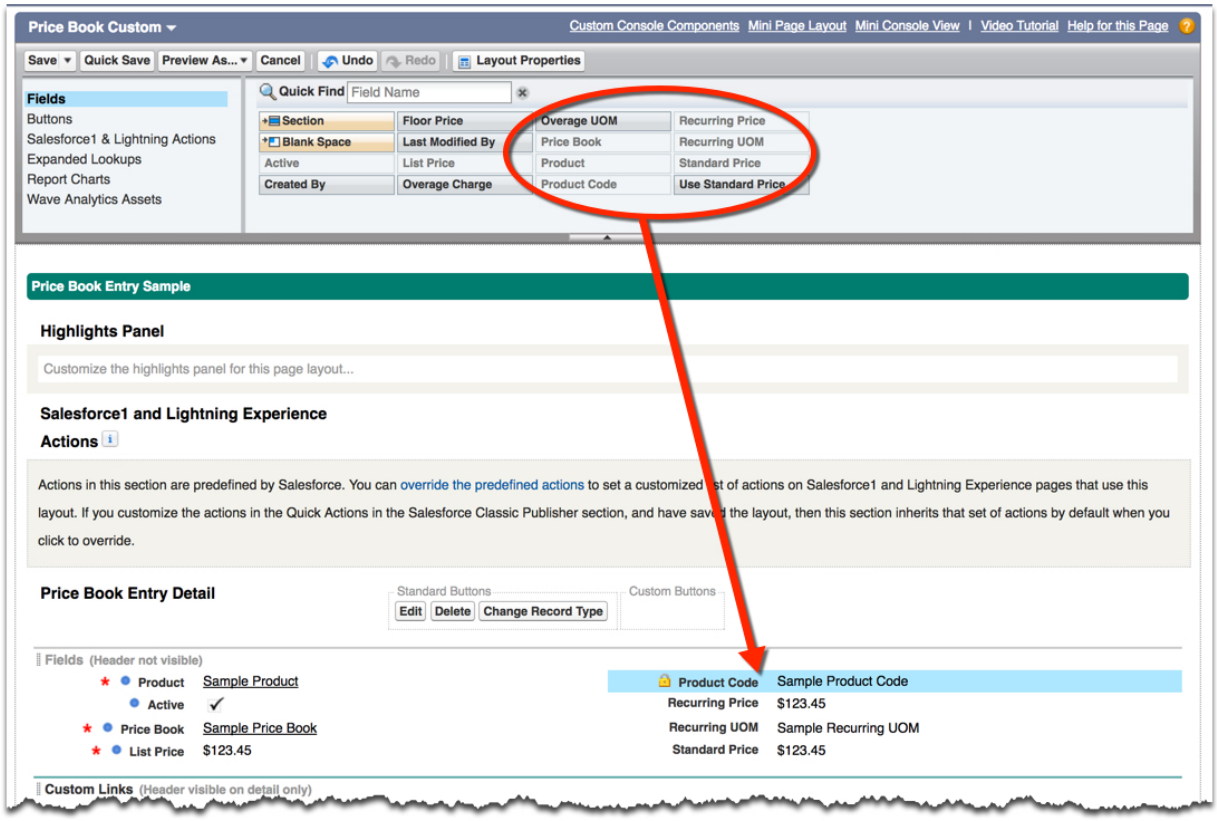
1. From Setup, click **Customize**.
2. Click the object to customize.
3. Click **Page Layouts**.



4. Click **New**.
5. In the **Page Layout Name** box, enter a name for the layout.



6. Click **Save**.
7. From the palette, drag Vlocity custom fields—for example, **Product Code**, **Recurring Price**, **Recurring UOM**, **Standard Price**—to the **Fields** section.

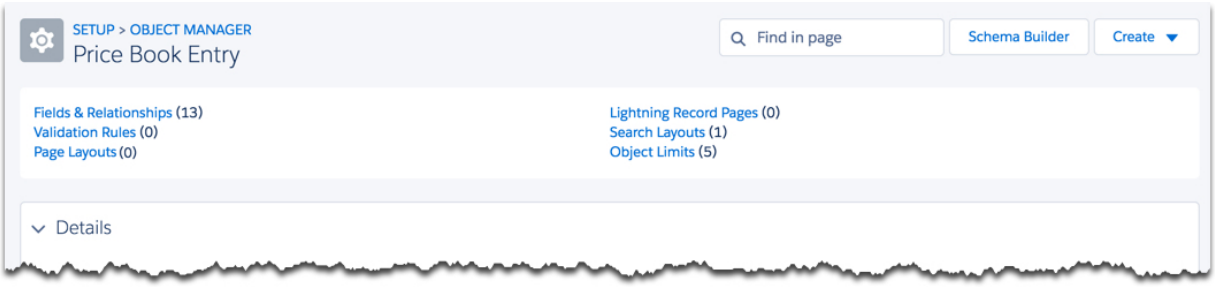


8. Modify any other sections as necessary.
9. Click **Save**.

Create a New Page Layout in Lightning Experience

To create a new page layout in Salesforce Lightning Experience:

1. From Setup, click the **Quick Find** box, enter **`Object`** and click **Object Manager**.
2. Click the object for which you will change the page layout.



3. Click **Page Layouts**.
4. Follow steps 4 through 7 in the Salesforce Classic instructions.

Installing Cards, Templates, and OmniScripts

Vlocity recommends that you install all base cards, templates, and OmniScripts that are in your org. There is no negative impact on your environment and if you plan to use these other products in the future, you will have access to them automatically.

WARNING

If you customize any of your cards, templates, and OmniScripts, you must give them a new, unique name to prevent them from being overwritten when you refresh them for the latest release.

1. From the App Launcher, click **Vlocity Cards**.
2. Click the **Additional Actions** arrow in the top-right and install and activate all cards shown.
   The **Select Items to Import** dialog box opens.
3. Ensure that all items are selected. Click **Next**.
4. Review the items to import. Click **Next**.
5. Ignore the **Import Error** for eSignature.
6. When the import is complete, click **Activate Now**.
   The **Select Items to Activate** dialog box opens.

   IMPORTANT

   The latest version of the card, template or layout is automatically selected while the previous version is still active. If previous versions are still active, a yellow triangle icon appears in the Activation dialog and the new versions are not selected. You may have to scroll down to the bottom of the Activation dialog box to see all the yellow triangle icons. To solve this issue, click **Select All** or check the checkbox for each template version displaying the warning, then click Next. The previous versions are deactivated and the new versions are activated.

   If you deactivate any previous active versions of cards and layouts, including customized cards and layouts, clicking **Select All** will overwrite your customized versions with the new, default versions. To ensure your custom cards and layouts and not deactivated and overwritten, rename any default cards and layouts you plan to customize, deselect any newly-installed default versions of the cards and layouts, and select your custom cards and layouts for activation.
7. Ensure that all items are selected. Click **Next** and click **Done**.

8. Refresh your browser cache if necessary.
9. From the App Launcher, click **Vlocity Templates** and repeat the above procedure for all Vlocity templates.
10. If you are installing or upgrading CME Fall '19 or later and if a DataPack contains Lightning Web Components (LWC)-enabled OmniScripts and Cards, go to OmniScripts/Card Designer for each one and click **LWC Preview** to deploy the Lightning Web Component. You must do this after upgrading in order to use the latest LWC-enabled Omniscripts. For full instructions, see Create an LWC OmniScript.

Billing Management

Vlocity Communications, Media, and Energy does not include a billing engine, but Vlocity Communications, Media, and Energy can access, report, and analyze billing data. The Vlocity Billing Management module aggregates billing data across multiple billers on a schedule. For example, you can call the information up when a call comes in or schedule a monthly data load. Reporting can store bills, bill line items, and DataRaptor data.

You can create new payment methods and payment types. A payment method is how a customer compensates a seller for products and services. A payment type is a category to which a payment method belongs.

Billing Management Overview

Vlocity does not include a billing engine, but Vlocity can access, report, and analyze billing data. Prebuilt processes and the Vlocity data model interact with billing data. The Vlocity Billing Management module, not an engine, aggregates billing data across multiple billers. Billing Management aggregates billing data on a schedule.

For example, you can call the information up when a call comes in or schedule a monthly data load. Reporting can store bill, bill line items, and DataRaptor data.

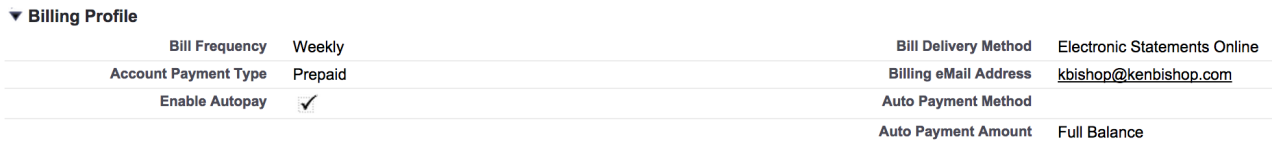
Communications-specific objects and the data model extend the Salesforce Case object to support billing disputes. Billing Management includes billing reports and dashboards, billing and usage interfaces, and data maintenance service.

You can manage billing directly from a Billing Aggregator, Billing, Business, or Consumer Account record. Statements, statement line items, and account balances are imported from your company's billing system. View comprehensive billing and usage reports. Add payment methods and create payment adjustments directly from the account record to be exported to the billing system for processing.

Neither Salesforce nor Vlocity are billing systems, but Billing Management retrieves specific information from other systems. This enables the customer service representative to handle billing questions. Representatives can make payments and adjustments. System administrators can define payment and adjustment methods and options.

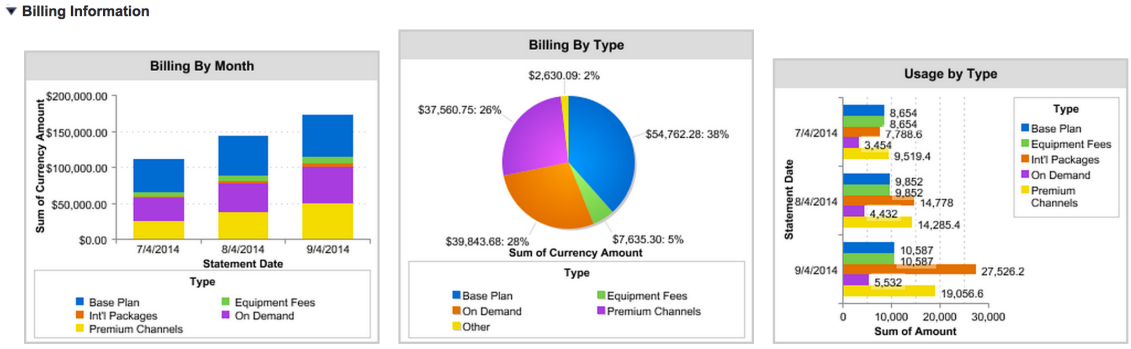
Billing Profile

The Billing Profile section includes all billing information for a customer, including payment methods, billing methods, and autopay options.



Billing Information

The Billing and Usage Information Visualforce page displays a variety of billing and usage reports, depending on the account type. In the example below, the image shows a graph of an account's billing by month, a pie chart of billing by type, and a graph of usage by type.



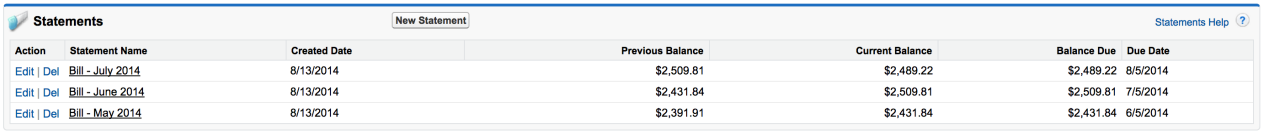
As of Yesterday at 7:20 PM

As of Yesterday at 7:20 PM

As of Yesterday at 7:20 PM

Statements

The Statements related list displays recent statements, imported from your company's billing system.

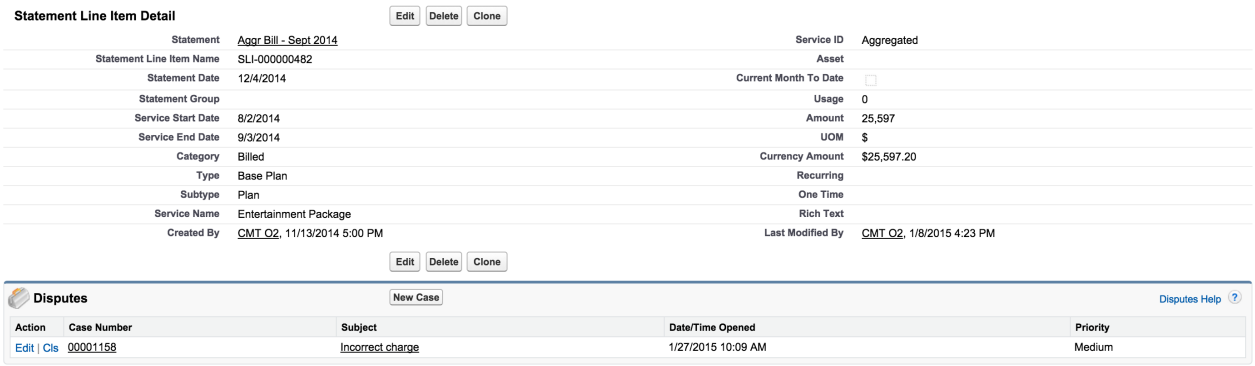


Statements — New Statement — Statements Help

Action	Statement Name	Created Date	Previous Balance	Current Balance	Balance Due	Due Date
Edit | Del	Bill - July 2014	8/13/2014	$2,509.81	$2,489.22	$2,489.22	8/5/2014
Edit | Del	Bill - June 2014	8/13/2014	$2,431.84	$2,509.81	$2,509.81	7/5/2014
Edit | Del	Bill - May 2014	8/13/2014	$2,391.91	$2,431.84	$2,431.84	6/5/2014

Click a statement to open the Statement record.

Each statement has statement line items, which display details for a product or service.



Statement Line Item Detail — Edit | Delete | Clone

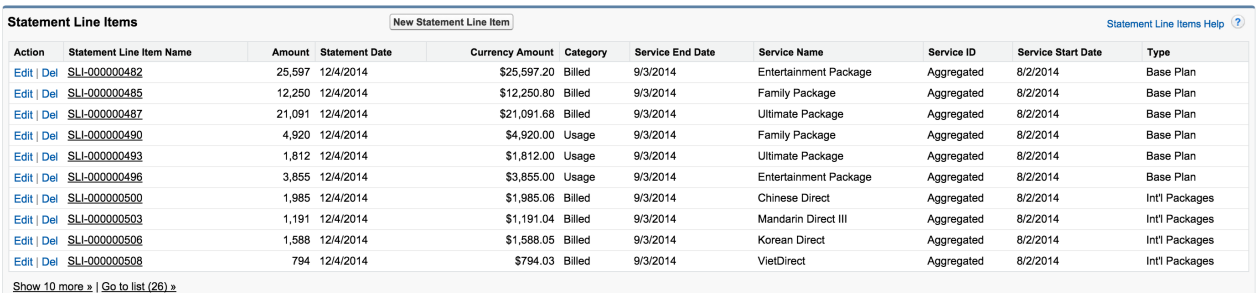
Statement	Aggr Bill - Sept 2014	Service ID	Aggregated
Statement Line Item Name	SLI-000000482	Asset	
Statement Date	12/4/2014	Current Month To Date	
Statement Group		Usage	0
Service Start Date	8/2/2014	Amount	25,597
Service End Date	9/3/2014	UOM	$
Category	Billed	Currency Amount	$25,597.20
Type	Base Plan	Recurring	
Subtype	Plan	One Time	
Service Name	Entertainment Package	Rich Text	
Created By	CMT O2, 11/13/2014 5:00 PM	Last Modified By	CMT O2, 1/8/2015 4:23 PM

Edit | Delete | Clone

Disputes — New Case — Disputes Help

Action	Case Number	Subject	Date/Time Opened	Priority
Edit | Cls	00001158	Incorrect charge	1/27/2015 10:09 AM	Medium

Any billing or service disputes related to the line item appear on the Statement Line Item record.



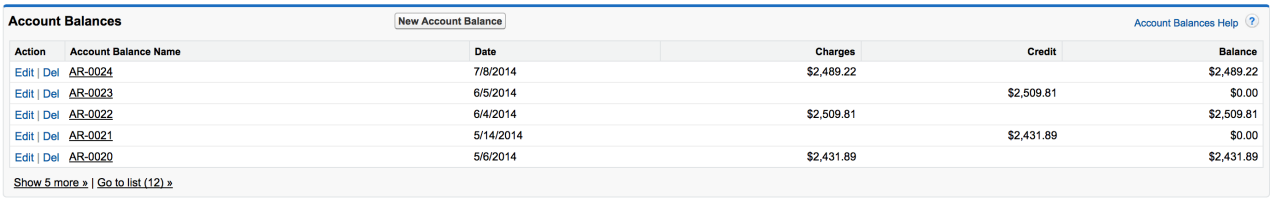
Statement Line Items — New Statement Line Item — Statement Line Items Help

Action	Statement Line Item Name	Amount	Statement Date	Currency Amount	Category	Service End Date	Service Name	Service ID	Service Start Date	Type
Edit | Del	SLI-000000482	25,597	12/4/2014	$25,597.20	Billed	9/3/2014	Entertainment Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000485	12,250	12/4/2014	$12,250.80	Billed	9/3/2014	Family Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000487	21,091	12/4/2014	$21,091.68	Billed	9/3/2014	Ultimate Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000490	4,920	12/4/2014	$4,920.00	Usage	9/3/2014	Family Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000493	1,812	12/4/2014	$1,812.00	Usage	9/3/2014	Ultimate Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000496	3,855	12/4/2014	$3,855.00	Usage	9/3/2014	Entertainment Package	Aggregated	8/2/2014	Base Plan
Edit | Del	SLI-000000500	1,985	12/4/2014	$1,985.06	Billed	9/3/2014	Chinese Direct	Aggregated	8/2/2014	Int'l Packages
Edit | Del	SLI-000000503	1,191	12/4/2014	$1,191.04	Billed	9/3/2014	Mandarin Direct III	Aggregated	8/2/2014	Int'l Packages
Edit | Del	SLI-000000506	1,588	12/4/2014	$1,588.05	Billed	9/3/2014	Korean Direct	Aggregated	8/2/2014	Int'l Packages
Edit | Del	SLI-000000508	794	12/4/2014	$794.03	Billed	9/3/2014	VietDirect	Aggregated	8/2/2014	Int'l Packages

Show 10 more » | Go to list (26) »

Account Balances

The Account Balances related list displays a running balance for the account imported from your company's billing system.



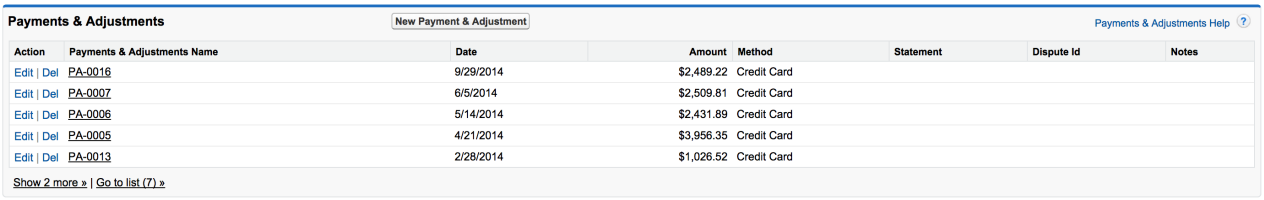
Account Balances — New Account Balance — Account Balances Help

Action	Account Balance Name	Date	Charges	Credit	Balance
Edit | Del	AR-0024	7/8/2014	$2,489.22		$2,489.22
Edit | Del	AR-0023	6/5/2014		$2,509.81	$0.00
Edit | Del	AR-0022	6/4/2014	$2,509.81		$2,509.81
Edit | Del	AR-0021	5/14/2014		$2,431.89	$0.00
Edit | Del	AR-0020	5/6/2014	$2,431.89		$2,431.89

Show 5 more » | Go to list (12) »

Payments and Adjustments

The Payments & Adjustments related list displays recent payments and customer service or other system adjustments. You can enter a new payment method or adjustment. The payment or adjustment is sent to your company's billing system for processing.



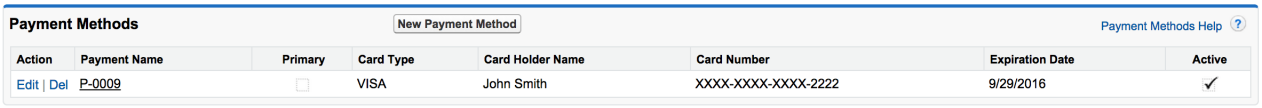
Payments & Adjustments — New Payment & Adjustment — Payments & Adjustments Help

Action	Payments & Adjustments Name	Date	Amount	Method	Statement	Dispute Id	Notes
Edit | Del	PA-0016	9/29/2014	$2,489.22	Credit Card			
Edit | Del	PA-0007	6/5/2014	$2,509.81	Credit Card			
Edit | Del	PA-0006	5/14/2014	$2,431.89	Credit Card			
Edit | Del	PA-0005	4/21/2014	$3,956.35	Credit Card			
Edit | Del	PA-0013	2/28/2014	$1,026.52	Credit Card			

Show 2 more » | Go to list (7) »

Payment Methods

The Payment Methods related list displays all payment methods on file. You can also add new payment methods or remove payment methods that are no longer in use. You can add a new credit card, bank account, or other payment method to the account record.



Payment Methods — New Payment Method — Payment Methods Help

Action	Payment Name	Primary	Card Type	Card Holder Name	Card Number	Expiration Date	Active
Edit | Del	P-0009	☐	VISA	John Smith	XXXX-XXXX-XXXX-2222	9/29/2016	✓

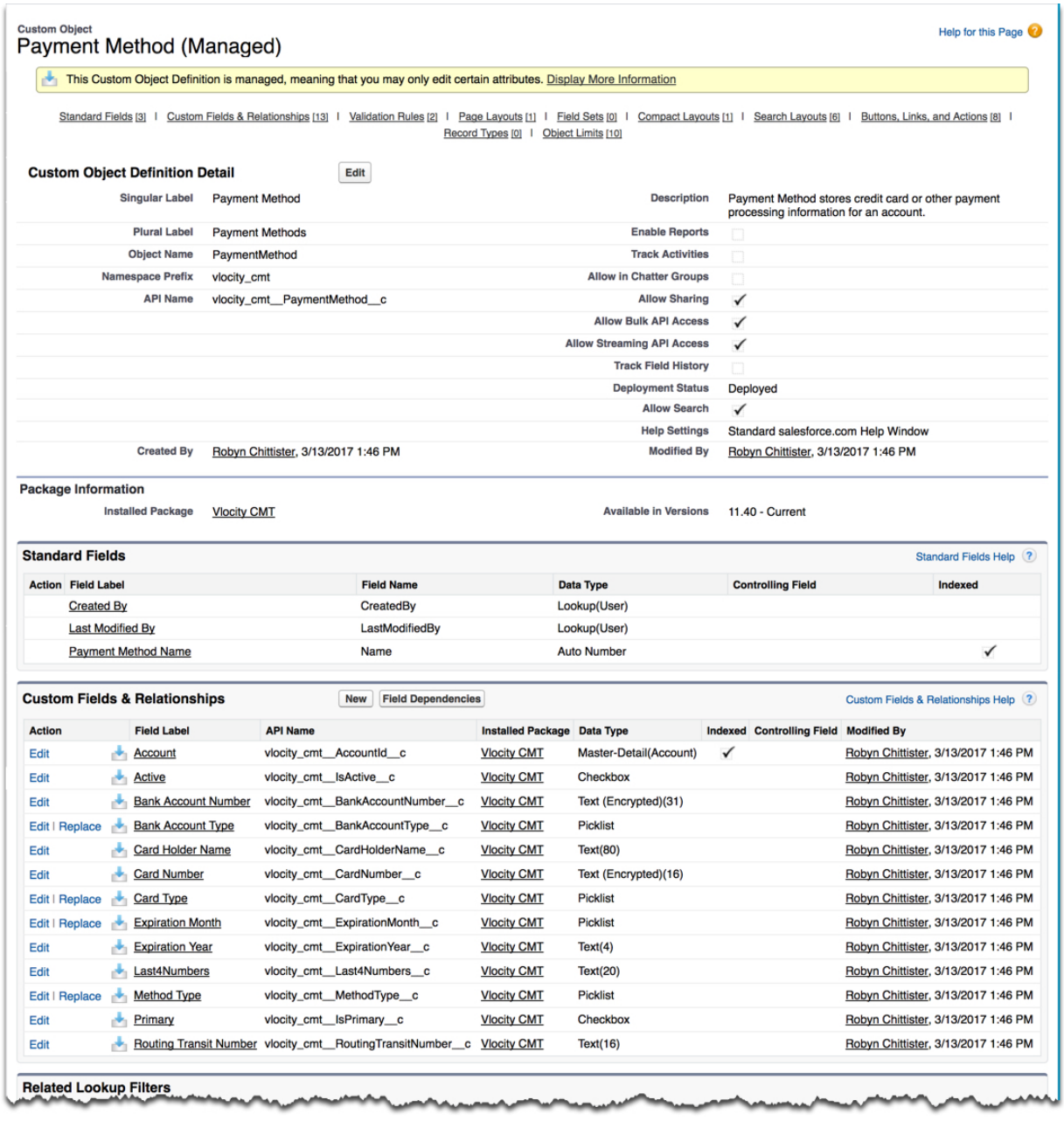
Configure Payment Methods

The Payment Method object stores credit card or other payment processing information for an account. You can configure the card and bank account types available for selection when adding new payment methods.

Before deploying Vlocity, set up the picklists in the Payment Methods and Payment Adjustment objects. For more information, see Add or Edit Picklist Values in the Salesforce Help.

To configure payment methods:

1. From Setup, in the **Quick Find** box, enter **`Objects`**.
2. Click **Objects**.
3. Click **Payment Method**.



Custom Object

Payment Method (Managed)

Help for this Page

This Custom Object Definition is managed, meaning that you may only edit certain attributes. Display More Information

Standard Fields [3] | Custom Fields & Relationships [13] | Validation Rules [2] | Page Layouts [1] | Field Sets [0] | Compact Layouts [1] | Search Layouts [6] | Buttons, Links, and Actions [8] | Record Types [0] | Object Limits [10]

Custom Object Definition Detail Edit

Singular Label	Payment Method	Description	Payment Method stores credit card or other payment processing information for an account.
Plural Label	Payment Methods	Enable Reports	
Object Name	PaymentMethod	Track Activities	
Namespace Prefix	vlocity_cmt	Allow in Chatter Groups	
API Name	vlocity_cmt__PaymentMethod__c	Allow Sharing	✓
		Allow Bulk API Access	✓
		Allow Streaming API Access	✓
		Track Field History	
		Deployment Status	Deployed
		Allow Search	✓
		Help Settings	Standard salesforce.com Help Window
Created By	Robyn Chittister, 3/13/2017 1:46 PM	Modified By	Robyn Chittister, 3/13/2017 1:46 PM

Package Information

Installed Package	Vlocity CMT	Available in Versions	11.40 - Current

Standard Fields Standard Fields Help

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Last Modified By	LastModifiedBy	Lookup(User)		
	Payment Method Name	Name	Auto Number		✓

Custom Fields & Relationships New | Field Dependencies — Custom Fields & Relationships Help

Action	Field Label	API Name	Installed Package	Data Type	Indexed	Controlling Field	Modified By
Edit	Account	vlocity_cmt__AccountId__c	Vlocity CMT	Master-Detail(Account)	✓		Robyn Chittister, 3/13/2017 1:46 PM
Edit	Active	vlocity_cmt__IsActive__c	Vlocity CMT	Checkbox			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Bank Account Number	vlocity_cmt__BankAccountNumber__c	Vlocity CMT	Text (Encrypted)(31)			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Bank Account Type	vlocity_cmt__BankAccountType__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Card Holder Name	vlocity_cmt__CardHolderName__c	Vlocity CMT	Text(80)			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Card Number	vlocity_cmt__CardNumber__c	Vlocity CMT	Text (Encrypted)(16)			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Card Type	vlocity_cmt__CardType__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Expiration Month	vlocity_cmt__ExpirationMonth__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Expiration Year	vlocity_cmt__ExpirationYear__c	Vlocity CMT	Text(4)			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Last4Numbers	vlocity_cmt__Last4Numbers__c	Vlocity CMT	Text(20)			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Method Type	vlocity_cmt__MethodType__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Primary	vlocity_cmt__IsPrimary__c	Vlocity CMT	Checkbox			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Routing Transit Number	vlocity_cmt__RoutingTransitNumber__c	Vlocity CMT	Text(16)			Robyn Chittister, 3/13/2017 1:46 PM

Related Lookup Filters

The **Custom Fields & Relationships** related list contains the fields that specify information about payment methods, including:

- **Active** specifies if a payment method is active or inactive.
- **Bank Account Type** specifies the type of bank account from which a customer can make a payment.
- **Card Type** specifies the type of credit card the customer can use to make a payment.
- **Method Type** specifies the type of payment.
- **Primary** specifies if a payment method is the primary payment method.

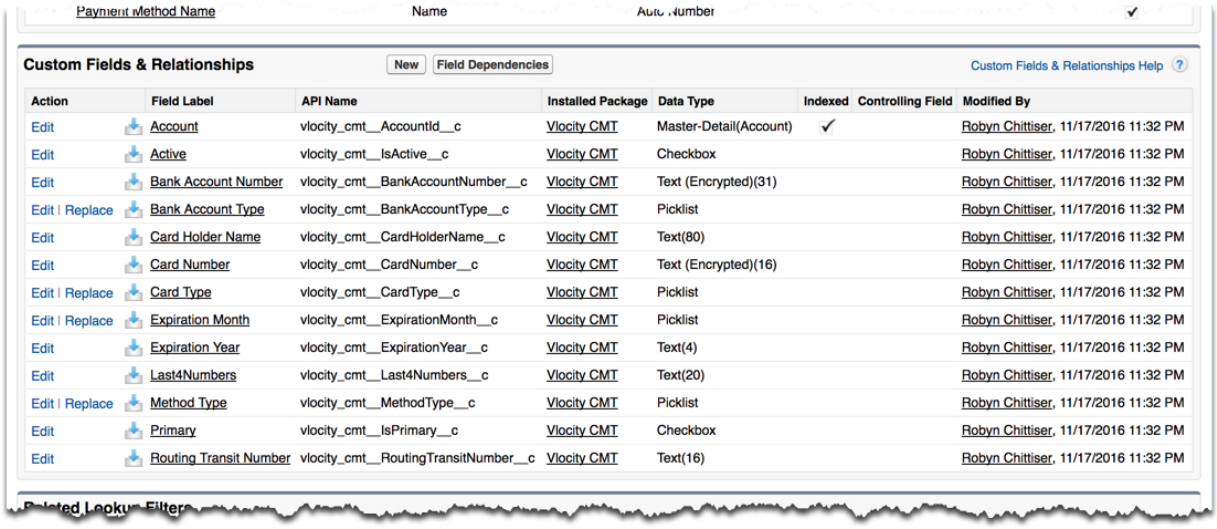
4. In the Custom Fields & Relationships related list, click the field label to edit its data.
5. Make any necessary changes.
6. Click **Save**.

Add a New Card Type

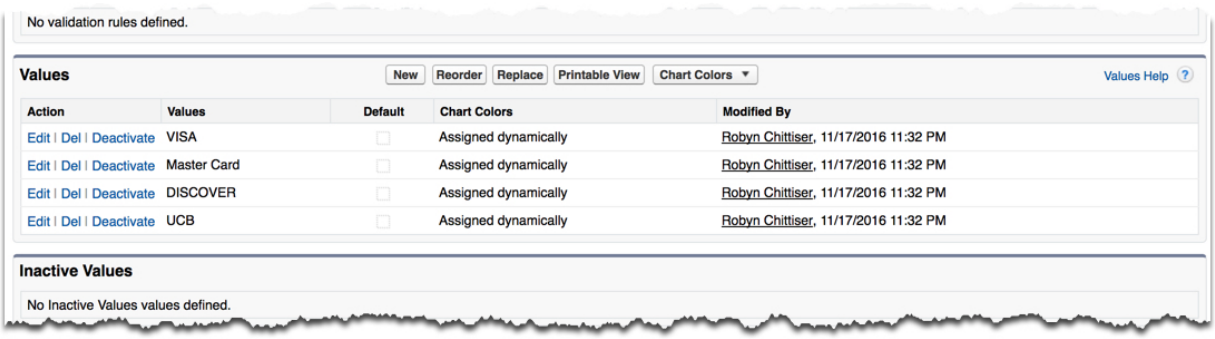
Add a card type on the Payment Method page.

To add a new card type:

1. On the Payment Method page, scroll down to the **Custom Fields & Relationships** related list.



2. Click **Card Type**.
3. On the Card Type page, scroll down to the **Values** related list.



4. Click **New**.
5. In the **Card Type** box, enter the appropriate information.
   Add one or more values—for example, **`American Express`**.
6. Click **Save**.

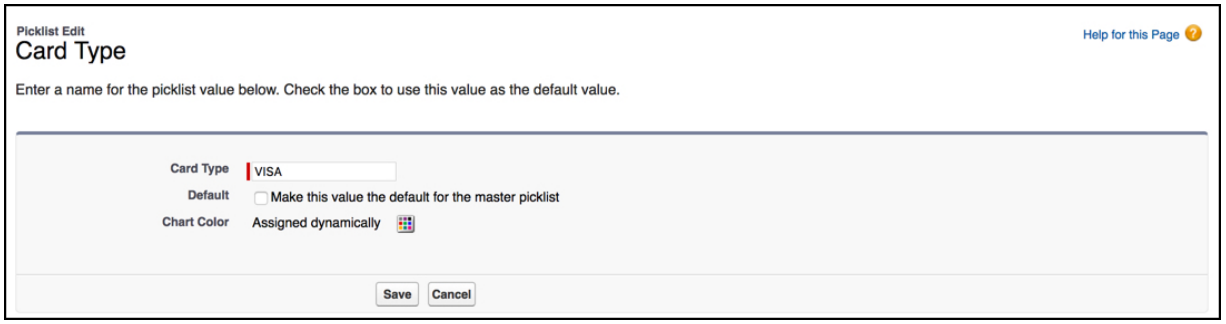
> **NOTE**
> You can add other payment methods as well. For example to add a new account payment method, in the **Custom Fields & Relationships** related list, click **Account**.

Edit a Card Type

Edit a card type on the Card Type page.

To edit a card type:

1. On the Card Type page, scroll down to the **Values** related list.
2. Next to one of the existing values, click **Edit**.



3. Edit the following information:
   - The **Card Type** is the card type as it will appear in the picklist.
   - The **Default** check box indicates if this is the default value for the master picklist.
   - The **Chart Color** determines how the card appears in charts.

4. Click **Save**.

Delete a Card Type

Delete a card type from the Card Type page.

To delete a card type:

1. On the Card Type page, scroll down to the **Values** related list.
2. Next to one of the existing values, click **Del**.

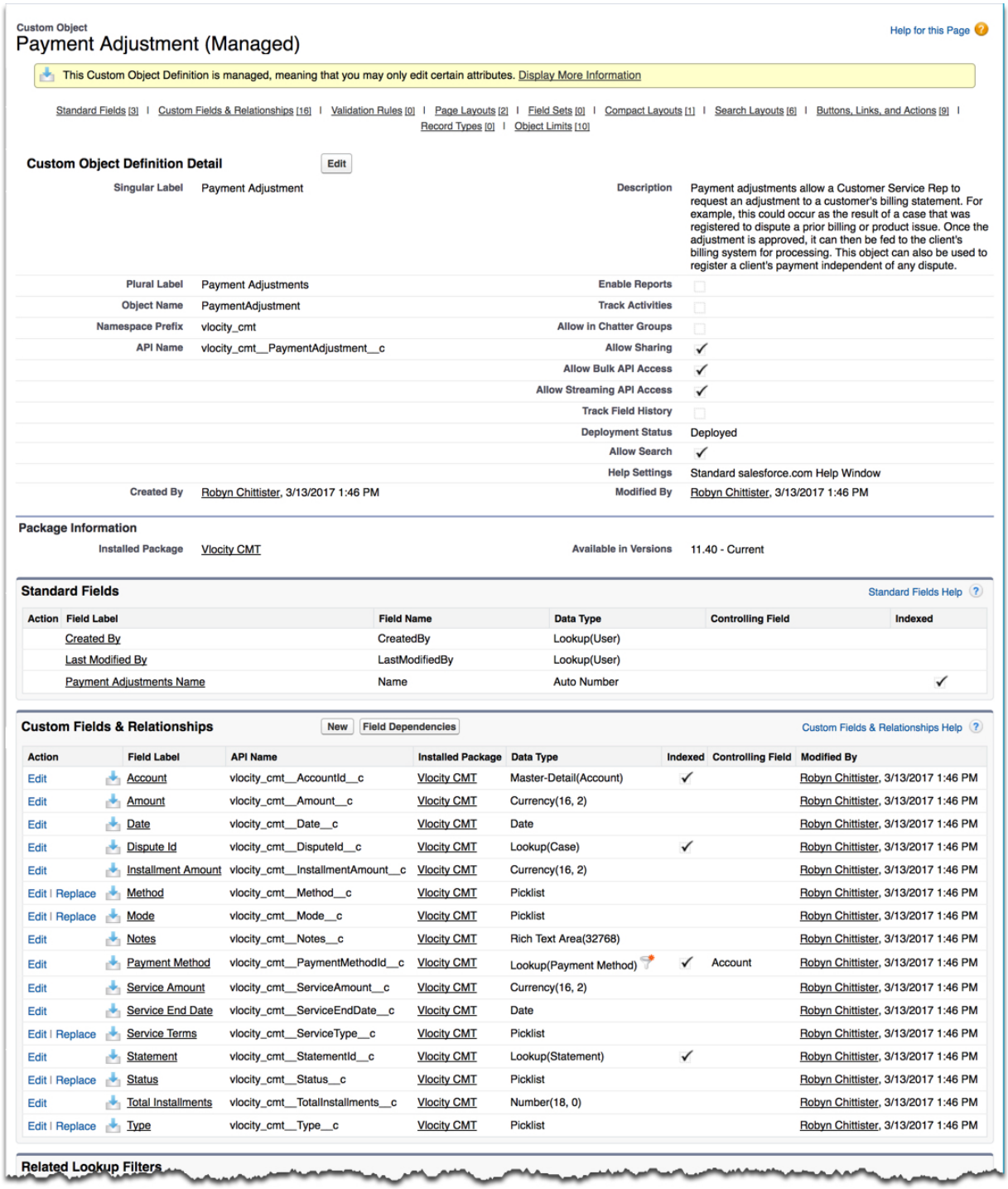
Configure Payment Adjustments

Payment adjustments allow a customer service representative to request an adjustment to a customer's billing statement, for example, when a customer disputes a prior billing or has a product issue. After the adjustment is approved, it can be sent to the client's billing system for processing. You can also use the Payment Adjustment object to register a client's payment independent of any dispute.

Before deploying Vlocity, set up the picklists in the Payment Methods and Payment Adjustment objects. For more information, see Add or Edit Picklist Values in the Salesforce Help.

To configure payment adjustment methods:

1. From Setup, in the **Quick Find** box, enter **`Objects`**.
2. Click **Objects**.
3. Click **Payment Adjustment**.



Custom Object

Payment Adjustment (Managed)

Help for this Page

This Custom Object Definition is managed, meaning that you may only edit certain attributes. Display More Information

Standard Fields [3] | Custom Fields & Relationships [16] | Validation Rules [0] | Page Layouts [2] | Field Sets [0] | Compact Layouts [1] | Search Layouts [6] | Buttons, Links, and Actions [9] | Record Types [0] | Object Limits [10]

Custom Object Definition Detail Edit

Singular Label	Payment Adjustment	Description	Payment adjustments allow a Customer Service Rep to request an adjustment to a customer's billing statement. For example, this could occur as the result of a case that was registered to dispute a prior billing or product issue. Once the adjustment is approved, it can then be fed to the client's billing system for processing. This object can also be used to register a client's payment independent of any dispute.
Plural Label	Payment Adjustments	Enable Reports	
Object Name	PaymentAdjustment	Track Activities	
Namespace Prefix	vlocity_cmt	Allow in Chatter Groups	
API Name	vlocity_cmt__PaymentAdjustment__c	Allow Sharing	✓
		Allow Bulk API Access	✓
		Allow Streaming API Access	✓
		Track Field History	
		Deployment Status	Deployed
		Allow Search	✓
		Help Settings	Standard salesforce.com Help Window
Created By	Robyn Chittister, 3/13/2017 1:46 PM	Modified By	Robyn Chittister, 3/13/2017 1:46 PM

Package Information

Installed Package	Vlocity CMT	Available in Versions	11.40 - Current

Standard Fields Standard Fields Help

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Last Modified By	LastModifiedBy	Lookup(User)		
	Payment Adjustments Name	Name	Auto Number		✓

Custom Fields & Relationships New | Field Dependencies — Custom Fields & Relationships Help

Action	Field Label	API Name	Installed Package	Data Type	Indexed	Controlling Field	Modified By
Edit	Account	vlocity_cmt__AccountId__c	Vlocity CMT	Master-Detail(Account)	✓		Robyn Chittister, 3/13/2017 1:46 PM
Edit	Amount	vlocity_cmt__Amount__c	Vlocity CMT	Currency(16, 2)			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Date	vlocity_cmt__Date__c	Vlocity CMT	Date			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Dispute Id	vlocity_cmt__DisputeId__c	Vlocity CMT	Lookup(Case)	✓		Robyn Chittister, 3/13/2017 1:46 PM
Edit	Installment Amount	vlocity_cmt__InstallmentAmount__c	Vlocity CMT	Currency(16, 2)			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Method	vlocity_cmt__Method__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Mode	vlocity_cmt__Mode__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Notes	vlocity_cmt__Notes__c	Vlocity CMT	Rich Text Area(32768)			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Payment Method	vlocity_cmt__PaymentMethodId__c	Vlocity CMT	Lookup(Payment Method)	✓	Account	Robyn Chittister, 3/13/2017 1:46 PM
Edit	Service Amount	vlocity_cmt__ServiceAmount__c	Vlocity CMT	Currency(16, 2)			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Service End Date	vlocity_cmt__ServiceEndDate__c	Vlocity CMT	Date			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Service Terms	vlocity_cmt__ServiceType__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Statement	vlocity_cmt__StatementId__c	Vlocity CMT	Lookup(Statement)	✓		Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Status	vlocity_cmt__Status__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM
Edit	Total Installments	vlocity_cmt__TotalInstallments__c	Vlocity CMT	Number(18, 0)			Robyn Chittister, 3/13/2017 1:46 PM
Edit | Replace	Type	vlocity_cmt__Type__c	Vlocity CMT	Picklist			Robyn Chittister, 3/13/2017 1:46 PM

Related Lookup Filters

The **Custom Fields & Relationships** related list contains information about payment adjustments, including:

- **Dispute ID** is a lookup to a case.
- **Method** specifies the adjustment method.
- **Mode** specifies what the adjustment is adjusting.
- **Service Terms** specifies when or how often the adjustment takes place.
- **Statement** is a lookup to the customer's statement.
- **Status** specifies where the adjustment is in the approval process.
- **Type** specifies why the adjustment was given.

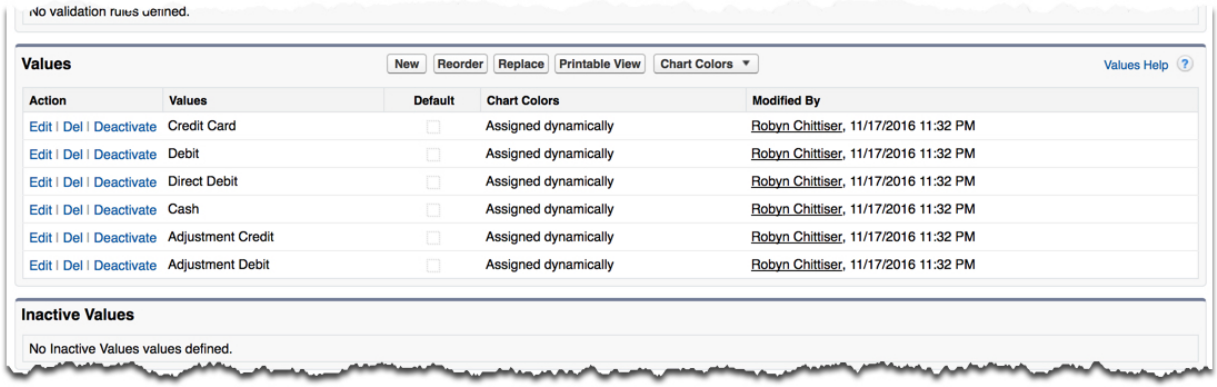
4. In the **Custom Fields & Relationships** related list, click the field label to edit its data.
5. Make any necessary changes.
6. Click **Save**.

Add a Payment Adjustment Method

Add a payment adjustment method from the Payment Adjustment object.

To add a payment adjustment method:

1. From Setup, in the **Quick Find** box, enter **`Objects`**.
2. Click **Objects**.
3. Click **Payment Adjustment**.
4. Scroll down to the **Custom Fields & Relationships** related list.
5. Click **Method**.
6. On the Method page, scroll down to the **Values** related list.



7. Click **New**.
8. In the **Method** box, enter the appropriate information.
9. Click **Save**.

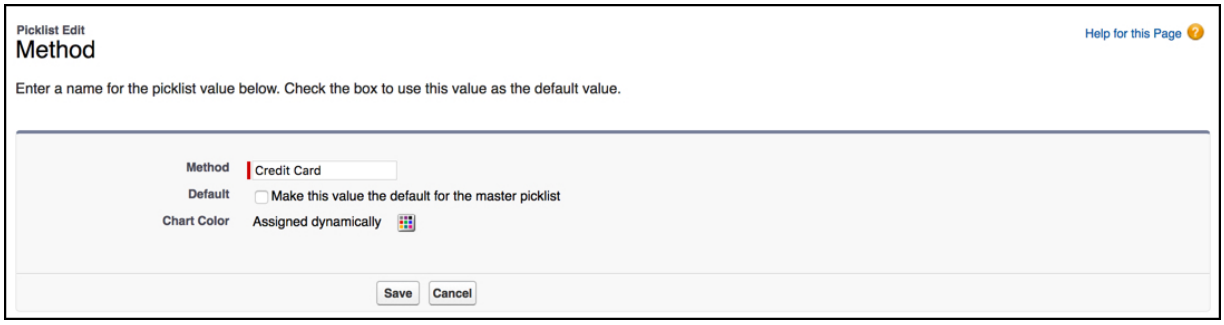
Edit a Payment Adjustment Method

Edit a payment adjustment method from the Payment Adjustment object.

To edit a payment adjustment method:

1. On the Method page, scroll down to the **Values** related list.

2. Next to one of the existing values, click **Edit**.



3. Edit the following information:
   - The **Method** is the adjustment method as it will appear in the picklist.
   - The **Default** check box indicates if this is the default value for the master picklist.
   - The **Chart Color** determines how the card appears in charts.
4. Click **Save**.

Delete a Payment Adjustment Method

Delete a payment adjustment method from the Payment Adjustment object.

To delete a payment adjustment method:

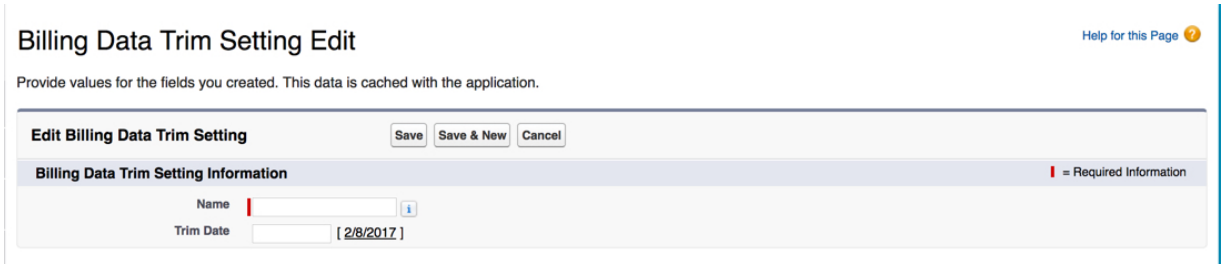
1. On the Method page, scroll down to the **Values** related list.
2. Next to one of the existing values, click **Del**.

Remove Billing Data

To prevent billing information from taking up too much storage in your org, use the Billing Data Trim setting to purge old billing statements.

To create a new Billing Data Trim Setting:

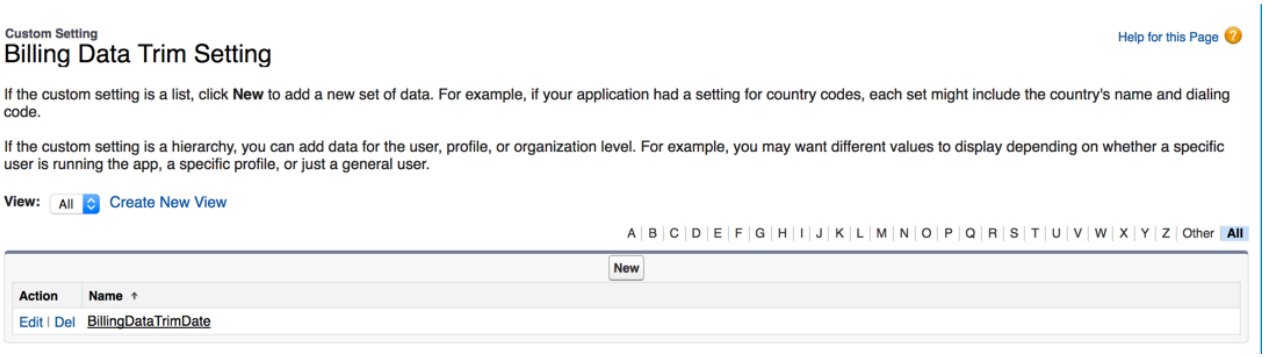
1. From Setup, in the **Quick Find** box, enter **`Custom Settings`**.
2. Click **Custom Settings**.
3. Next to Billing Data Trim Setting, click **Manage**.
4. Click **New**.



5. On the Billing Data Trim Setting Edit page, enter the following information:

- **Name** is the setting name.
- **Trim Date** is a cut-off date. All billing statements before this date are deleted from the system.

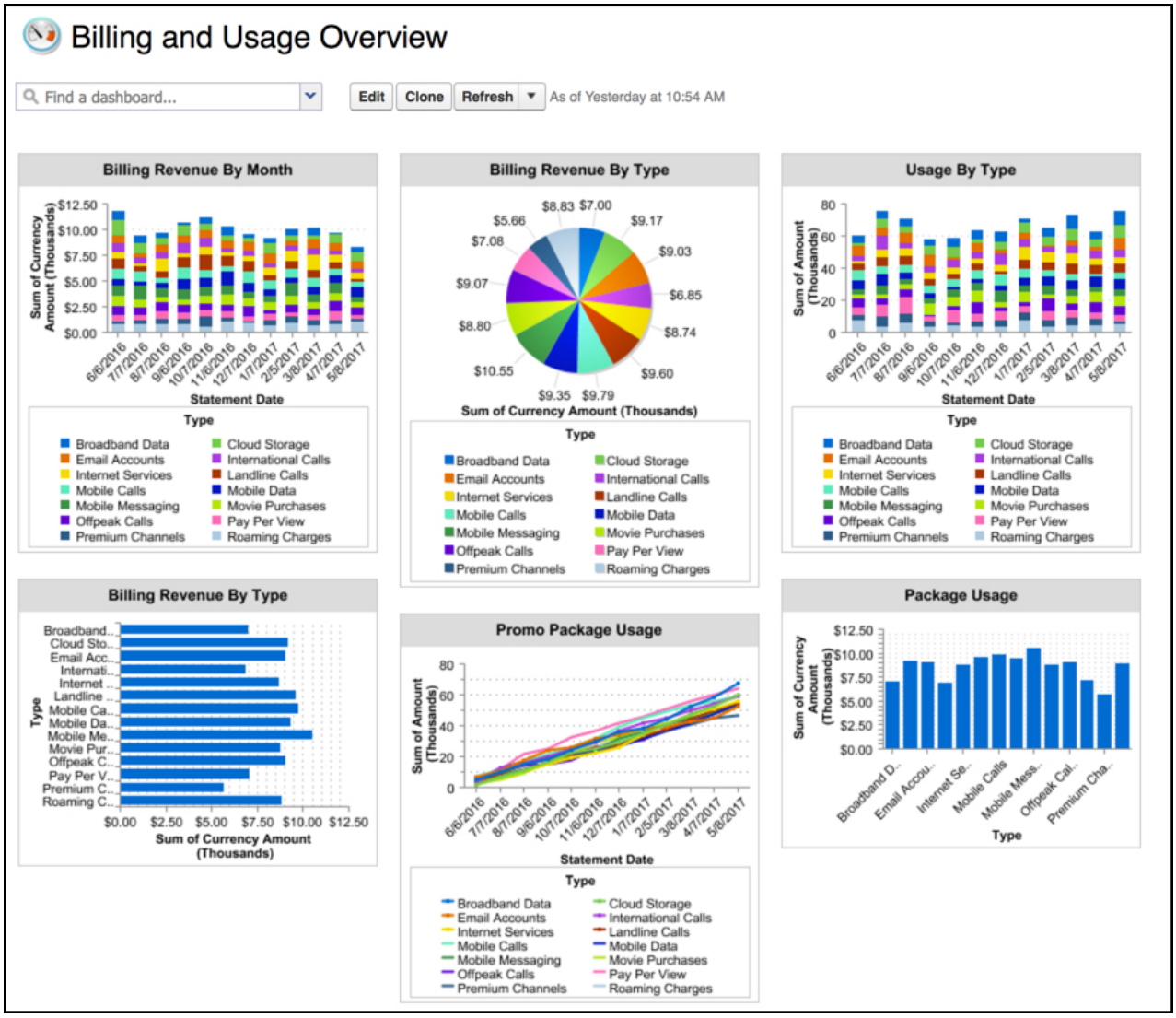
6. Click **Save**.



Run Billing Reports

Vlocity Communications, Media, and Energy includes some billing reports and usage reports, which appear in the Billing Information Visualforce page on an Account record. Reports on Billing Aggregator and Service Aggregator records roll up all usage and billing information from billing statements of child accounts. Customer service representatives can see billing information at a glance.

You configure these reports on the Reports tab. To modify a report, on the report page, click **Customize**. For more information, about the Salesforce Report Builder, see the Salesforce Help.



Forecasting

A forecast is an expression of expected revenue from sales based on the gross roll-up of a given set of probabilities. The task of forecasting is an imprecise art, as there are many unexpected items and factors that can cause shifts during the process. To achieve the best possible outcome, apply several forecasting methods.

Future planning is a key success factor for any business. While there are many variables that may affect a predicted business model, the forecasting process helps sales and administration staff convey an accurate plan for the sales cycle, from the pipeline to closed sales.

Collaborative and Customizable Forecast Methods

Salesforce provides two forecasting types you can use in the Salesforce environment: Collaborative Forecast and Customizable Forecast. Each type provides a different approach, but the goal of both types is to produce an accurate sales projection based on available data. In Salesforce, Opportunities and Schedules drive both types of forecasting. Vlocity supports and extends both forecasting types. However, if you do not use the Vlocity modifications, you can still use forecasting features from Salesforce.

Use Collaborative Forecast to predict and plan the sales cycle from the known pipeline to the closed sales stage, as well as to manage the sales expectations. For more information, see the Salesforce document, *Collaborative Forecasts Implementation Guide*.

Customizable Forecast is a flexible way to estimate how much revenue the organization can generate or how many items can be sold. You can only use Customizable Forecast with a Salesforce case. That is, to use Customizable Forecast, you must open a Salesforce case. For more information, see Customizable Forecasting Overview in the Salesforce Help.

Vlocity Forecasting Extensions

Vlocity supports and extends both of Salesforce's forecasting types: Collaborative Forecast and Customizable Forecast. The extension comes from the Vlocity CPQ ability to capture opportunity/quote and order information including both recurring and non-recurring revenue. You can include both of these revenue streams in forecast variables.

The `OpptyScheduleInterfaceDefaultImplementation` defines the specific approach to how recurring revenues are allocated into the forecast variable. You can choose to define an alternate or modified approach by developing a custom implementation. For more information, see the *Interfaces and Implementations Reference Guide*. For more information about forecast methods, see Forecasts Setup Overview in the Salesforce Help.

The Vlocity forecasting calculations differ from the Salesforce default forecasting calculations. The main differences between Vlocity and the Salesforce defaults are:

- Using Vlocity, you can create the schedules for an entire opportunity with one button. Using the Salesforce default, you must create schedules product by product.

- Salesforce divides the Opportunity.Amount (Total) evenly into the number of Opportunity months. Using Vlocity, the first month contains the one-time charges plus one instance of the recurring charge, while the remaining months contain just the recurring charge. The amount scheduling is based on the Number of Contracted Months in the Opportunity header.
- Vlocity forecasts according to months, while the Salesforce default forecasting can calculate by months, weeks, quarters, or years.

For more information about setting up forecasts, see the Salesforce document, Collaborative Forecasts Implementation Guide.

Here is an example for the Communications industry that shows the difference between the two scheduling types using Vlocity—an order for CATV service for a hotel chain. In the example, we assume that the length of the contract is 24 months.

Table 6. Example Forecast

Product	NRC (Install)	MRC	Qty (# rooms)
Regular Cable	10.00	30.00	100
HBO	5.00	40.00	100
Chinese Package	5.00	30.00	100
Total	20.00	100.00	100

- The non-recurring install charge is $20.00 * 100 = $2,000.
- The MRC (monthly recurring charge) is $100.00 * 100 per month, which is $10,000 per month or $240,000 over the life of the contract.

Using Collaborative Forecast, the whole amount of $242,000.00 would be forecast for the month of sale.

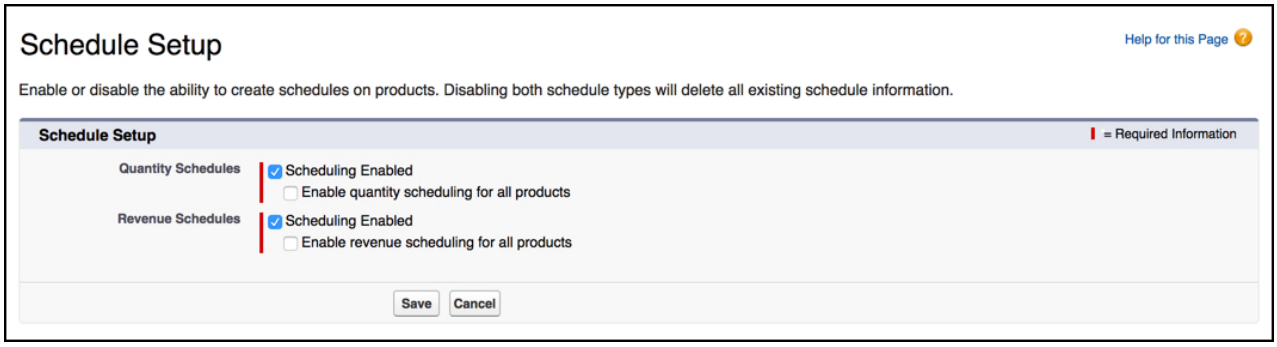
However, if you use Customizable Forecast and Vlocity `OpptyScheduleInterface`, the month of sale would include the amount of $22,000.00 (NRC + 1 MRC) and a $10,000.00 forecast for every month for the remaining 23 months of the contract. The forecast results are quite different.

Set Up Scheduling

Use scheduling to view how recurring revenue for a given opportunity is spread across a certain number of contracted months.

To enable scheduling:

1. From Setup, in the **Quick Find** box, enter **`Schedule`**.
2. Click **Product Schedules Settings**.



The number of contracted months drives the scheduling in a customizable forecasting environment. For example, if you have a 12-month contract for an opportunity, you get 12 columns on the schedule. If you have a 24-month contract you get 24 columns, and so on.

To see the schedule, on the Opportunity record detail page, expand the Products related list.

NOTE
The Products related list is not exposed out of the box. You can add it to the Opportunity page layout. For more information, see Page Layouts in the Salesforce Help.

Scheduling takes the one time total amount and spreads it over the chosen number of months. Scheduling applies to the technology the opportunity uses. However, it does not affect forecasting itself in the newer technology, but does affect it in the older technology.

NOTE
Do not click the **Re-establish** button.

If certain charges go up or down some months, you can enter comments to reflect these variations by clicking the **Edit** button.

If you try to create a schedule for an opportunity that already has a schedule, you will get an error message stating: The opportunity already has schedule data.

It is possible to remove an existing schedule by clicking the **Remove existing schedule data** button.

Setting Currency and Loyalty Codes

Loyalty pricing is a new feature in Vlocity Communications, Media, and Energy Winter '18. Two additional fields are included in price lists, **Currency Code** and **Loyalty Code**. By default, **Currency Code** is **USD** (US dollars) and **Loyalty Code** is **PTS** (points). You can modify these values as a part of the Salesforce global value sets. By default, for existing pricing lists, both the **Currency Code** and **Loyalty Code** aren't selected. However, the standalone pricing element requires the **Currency Code** and **Loyalty Code**. You must select the **Currency Code** and **Loyalty Code** to save any new and update all existing standalone pricing elements.

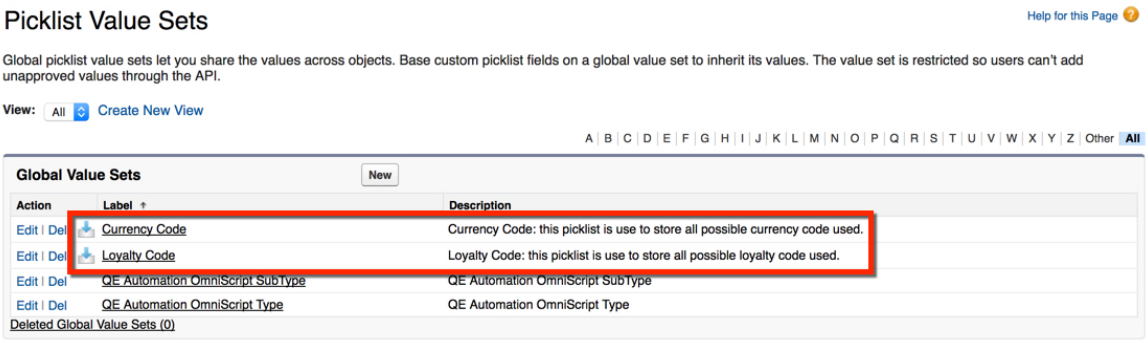
If you do not set a currency code for each price list, your standalone pricing elements will not have a currency code. This makes the pricing elements invalid. Products associated with invalid pricing elements are not displayed in the cart.

After you set the pricing and loyalty codes, you run the EPCPEUpdateCurrencyCode class, which sets the currency code for each standalone pricing element in the system.

You must perform these steps even if you are not using loyalty pricing.

To set the currency and loyalty codes:

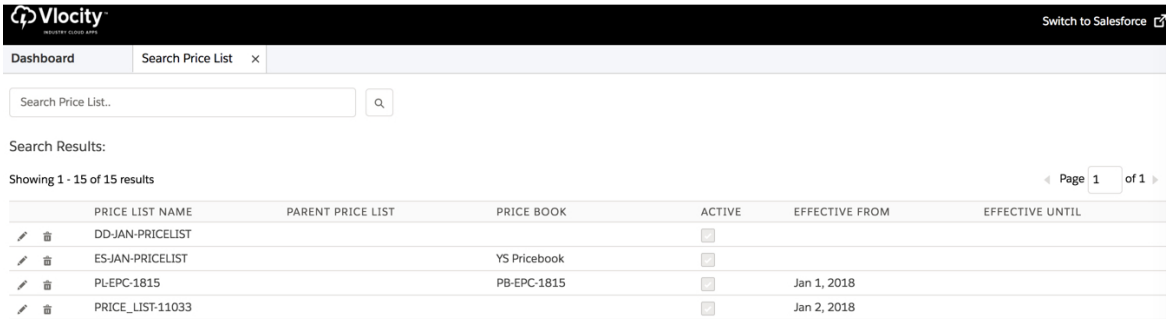
1. If you have not already done so, run the Install Default Vlocity Objects and Layouts batch job, located on the Vlocity CMT Administration tab, in the EPC batch jobs section. For more information, see Running Vlocity EPC Jobs for an Upgrade.
2. (Optional) Edit the Currency Code and Loyalty Code global value sets.
   a. From Setup, in the **Quick Find** box, enter **`Picklist`**.
   b. Click **Picklist Value Sets**.



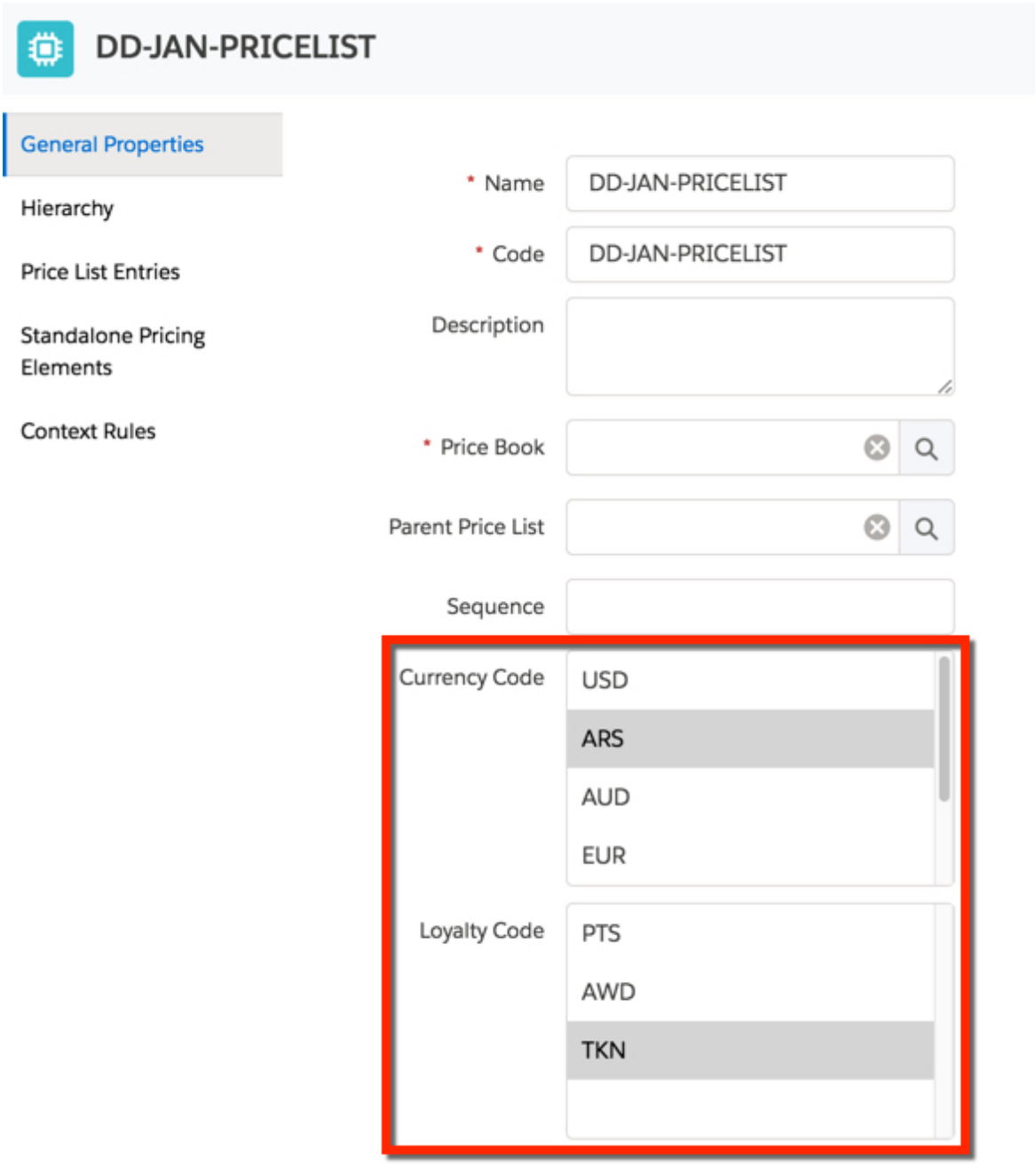
   c. Click **Currency Code**.
   d. In the **Values** related list, click **New**.
   The **Add Picklist Values Currency Code** page opens.
   e. Enter the currency codes.

f. Click **Save**.
g. Click **Back to List**.
h. Click **Loyalty Code**.
i. In the **Values** related list, click New.
The **Add Picklist Values Loyalty Code** page opens.
j. Enter the loyalty codes.
k. Click **Save**.

3. For each active price list, select a **Currency Code** value and a **Loyalty Code** value.
   a. In the Vlocity Product Console, next to Price List, click **Search**.
   b. Leave the **Search Price List** box blank and click **Search**.



   c. Next to the first active price list, click **Edit**.



d. In the **Currency Code** list, select the appropriate currency code.

NOTE
Although this is multiselect picklist, Vlocity supports only one currency code per price list. Select only one option.

e. In the **Loyalty Code** list, select the appropriate loyalty code.

NOTE
Although this is multiselect picklist, Vlocity supports only one loyalty code per price list. Select only one option.

f. Click **Save**.
g. Repeat steps c through f for each active price list.

Enable and Set Up Projects for Vlocity Product Designer

The Project feature tracks any changes that administrators make to EPC artifacts in the Vlocity Product Designer. To use Projects, you must enable and configure the feature, access, and record type.

To be able to use the Project feature, you need to complete the following procedures to set up the Projects object, enable the feature, create the Projects tab, and give users access to the Projects page and record type. For more information about using Projects, see Track Product Catalog Changes with Projects.

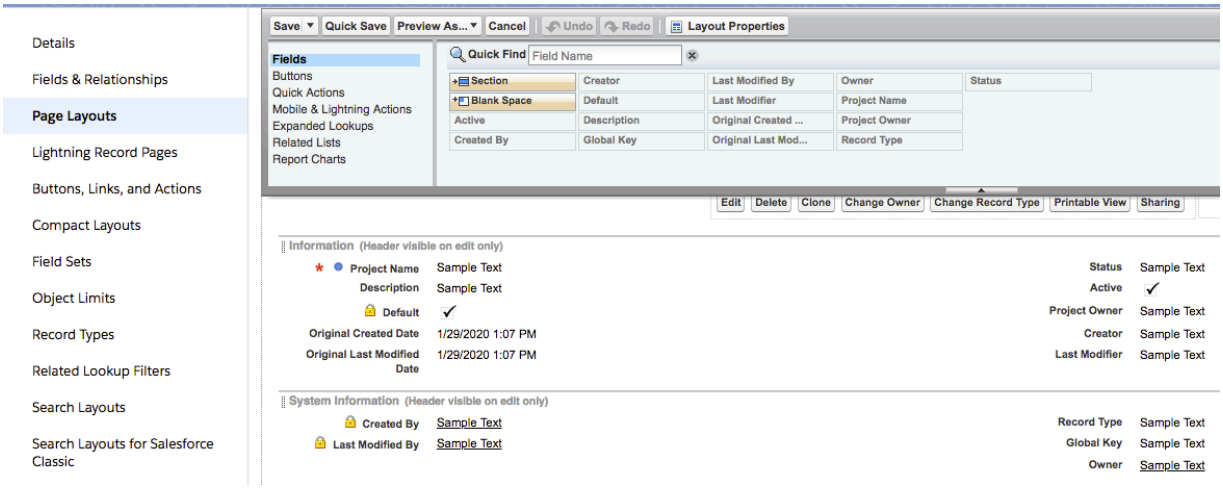
A default project is automatically created after the feature is enabled and when you create, update, or delete a product catalog object. This project is set as the default and is named Default Work Set for {user name} where {user name} is the full name of the logged-in user. The project has an item entry for the changed product.

To enable the EPC Project setting in Vlocity CMT Administration, see Configure Vlocity Product Designer.

Configure the Layout of the Project Custom Object

You must make the field information for Projects visible for the Project object.

1. From Setup, open **Object Manager**.
2. Click **Project** to open the object, and then click **Page Layouts**.
3. In the Page Layouts table, click **Project Layout**.
4. Drag the fields from the **Fields** list onto the Information and System Information sections of the layout as shown.

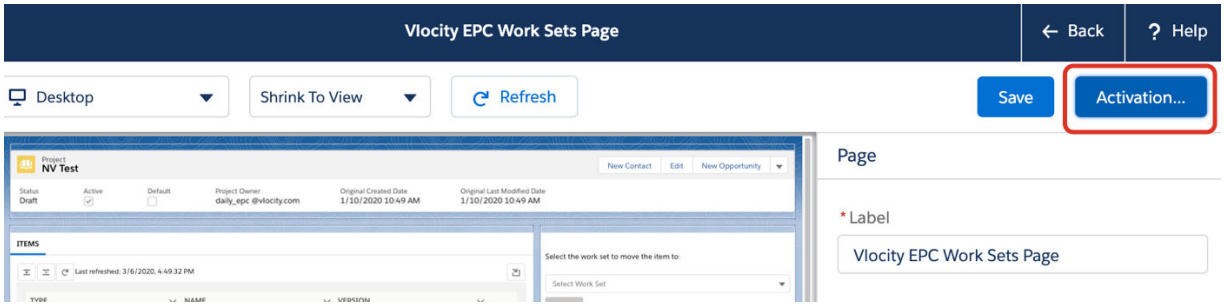


Project Fields for Information	Project Fields for System Information
• Project Name • Description • Default • Original Created Date • Original Last Modified Date • Status • Active • Project Owner • Creator • Last Modifier	• Created By • Last Modified By • Record Type • Global Key • Owner

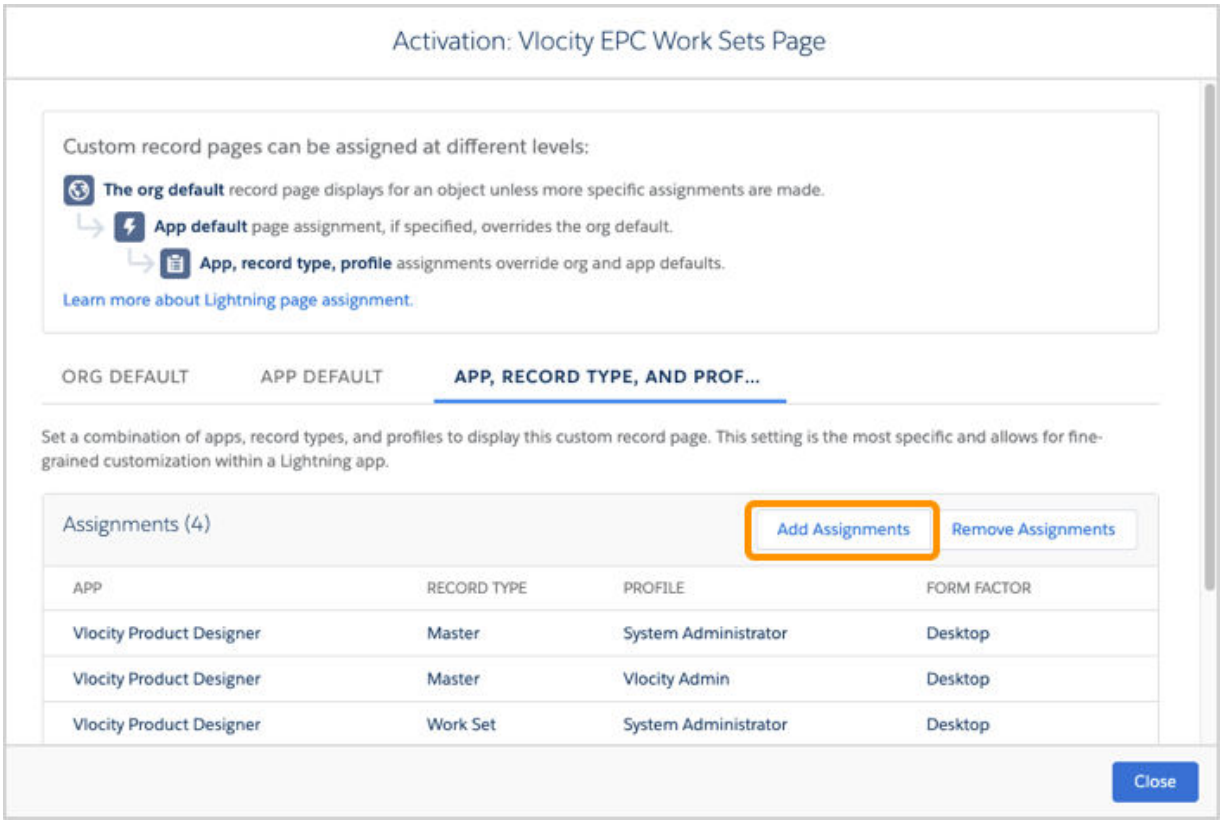
5. Click **Save**.

Enable the Project Lightning Record Page For Specific Profiles

1. From Setup, click **Object Manager**.
2. Click **Project** to open the object, and click **Lightning Record Pages**.
3. In the Lightning Record Pages table, click **Vlocity EPC Work Sets Page**.
4. Click **Edit**.
   Edit is not available for managed packages. Click on **View** instead.
5. Click **Activation**.



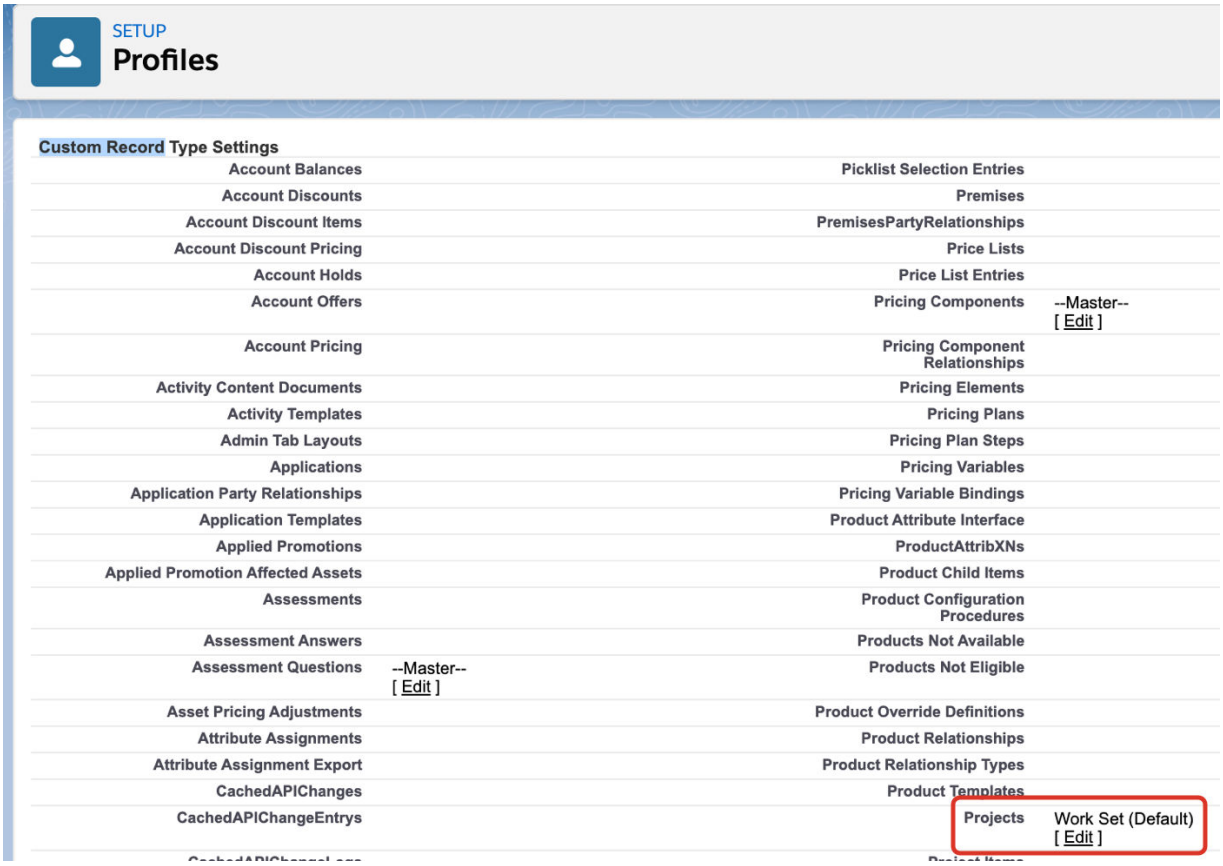
6. In the App, Record Type, and Profile tab, click **Add Assignments**.



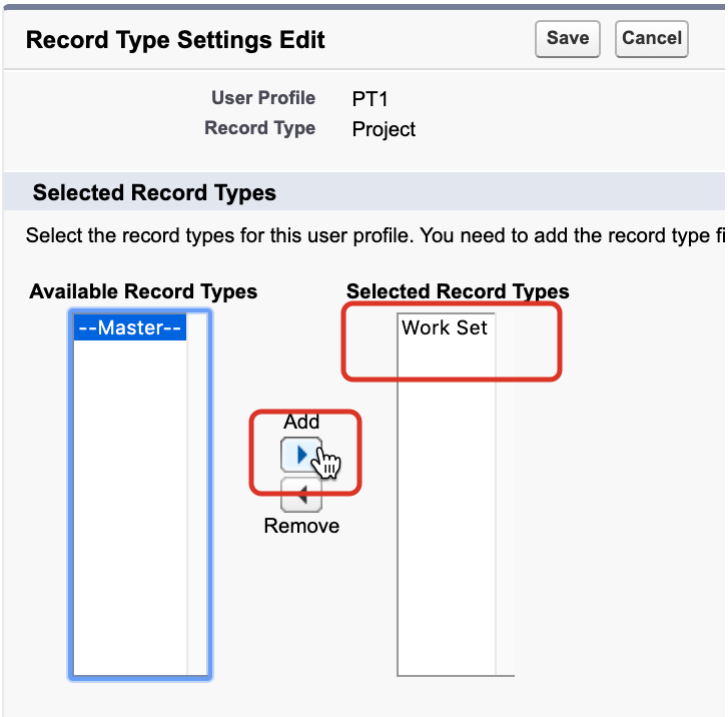
7. Select **Vlocity Product Console**, and click **Next**.
8. Select the **Work Set** and **Master** record types, and click **Next**.
9. Select your profiles, and click **Next**.
   Example: Vlocity Admin and System Administrator profiles.
10. Click **Save**.

Enable the Project Record Type for Specific User Profiles

1. From Setup, enter **`Profiles`** in the Quick Find box, and open **Profiles**.
2. Click the name of a profile to assign the Project record type to, such as System Administrator.
3. In the Custom Record Type Settings section of the profile details page, click **Edit** under Projects.



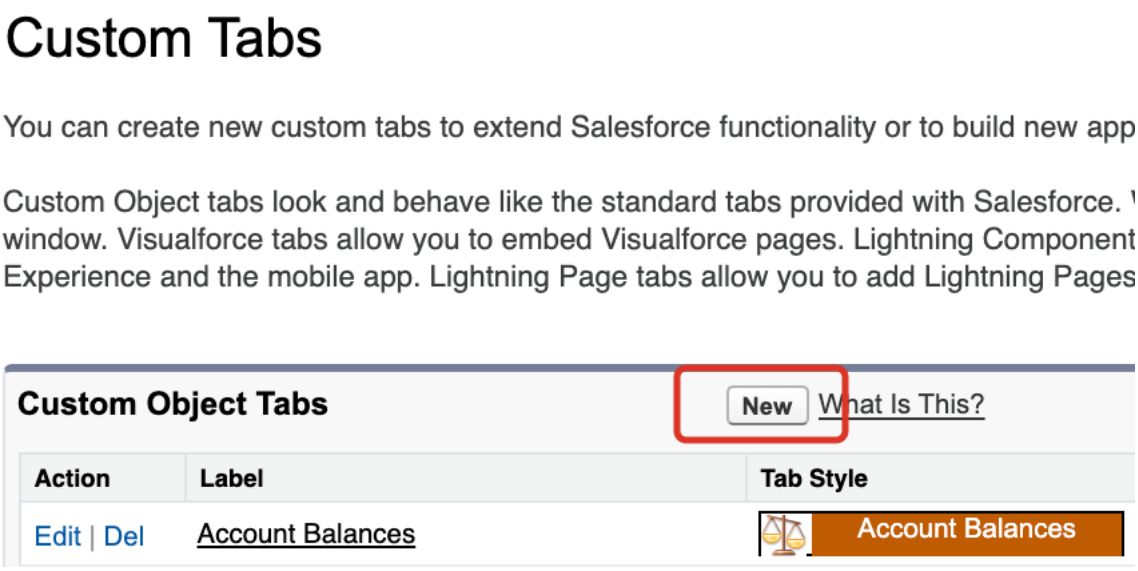
4. Select **Work Set** in the **Available Record Types** list and add it to the **Selected Record Types** list.



5. Click **Save**.
6. Repeat steps 3 - 5 for other profiles.

Create a Tab for the Project Custom Object

1. From Setup, enter **Tab** in the Quick Find box, and open **Tabs**.
2. In the Custom Tabs list, click **New**.



3. In the **Object** list, select **Project**.
   If more than one **Project** object is listed, select **Project (Installed Package: Vlocity CMT)**.
4. For the Tab Style, select **Form**, and click **Next**.
5. Accept the defaults, and click **Next**.
6. Click **Save**.

The tab will have the plural **Projects** as the label and will refer to the vlocity_cmt__Project__c object.

Multi-language Support for Shared Catalog

When communications service providers sell products in different regions, it's important that you translate significant product details into multiple languages. Translation helps sales agents find the things they need easily by making all the relevant details, such as properties, cross-sell, or upsell suggestions, available to them in their preferred language.

A UI-driven translation of Vlocity shared product catalog data in different locales enables sales agents and applications to view product data in local languages. The shared product catalog renders translated text for products, promotions, and properties in more than 18 different locales in CPQ, based on the user profile. You can also search and sort products and promotions by translated names in CPQ.

System administrators can use the multi-language support feature to configure a default language for a specific user or one or more languages that the user can choose from as a preferred language. The user can also change the default language in the user profile. The default language appears on the UI with language-specific field labels, as well as with translated metadata and business data.

At runtime in CPQ, the customer-preferred language has higher precedence than the default user-language. Similarly, if a language code is provided as part of a URL, as in eCom, then it has a higher precedence than the default user-language.

As an example, a customer initiates an order in French, through eCom, and completes the order by contacting a call center where a CSR has English as the default language. The CSR would see the Order data in French, provided the CSR locale/language is configured for French.

CPQ and Assets display products, promotions, and properties in the base language when translations aren't found.

With multi-language support, you can also enable users to configure the Shared Catalog objects and fields for translation. Users can determine the product data for translation based on the configured multi-language support object fields and languages. Users can then search and view the data for translation.

Vlocity supports the migration of translated product catalog data across Salesforce organizations.

You can also disable multi-language support at the organization level. After multi-language support is disabled, there's no translation in CPQ or Asset pages.

To set up multi-language support, complete the following tasks.

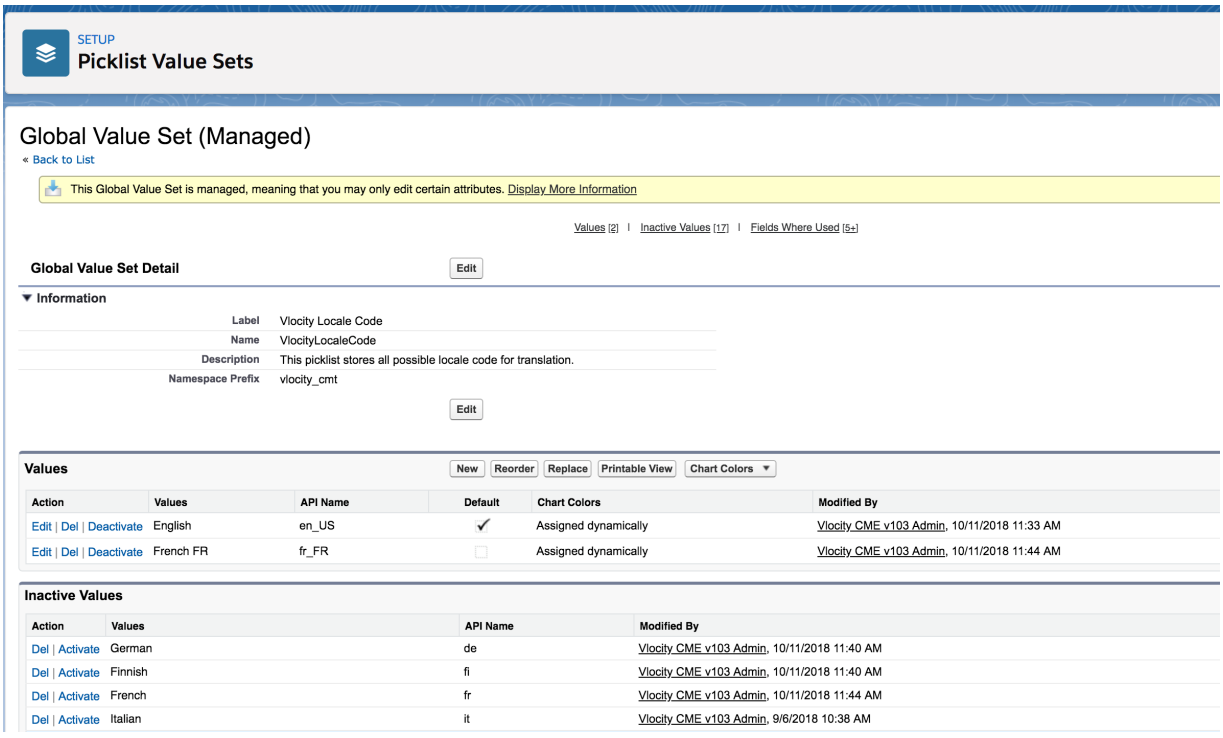
Task	Topic
Enable multi-language support in Vlocity CMT Administration.	Configuring Multi-Language Support for Shared Catalog
Activate and deactivate the languages that you need.	Configuring Multi-Language Support for Shared Catalog
Set the default user language.	Configuring Multi-Language Support for Shared Catalog
Configure the object fields for multi-language support.	Configuring Multi-Language Support for Shared Catalog

Task	Topic
Run the Create Translation job to extract product data (base language) for translation.	Configuring Multi-Language Support for Shared Catalog Create Translation Job
View the extracted untranslated product data.	Configuring Multi-Language Support for Shared Catalog
Translate the product data.	Set String Translations in Product Designer Modify String Translations in Vlocity Product Console
Run the Create Cache Translation Data job.	Configuring Multi-Language Support for Shared Catalog Create Cache Translation Data

Configuring Multi-Language Support for Shared Catalog

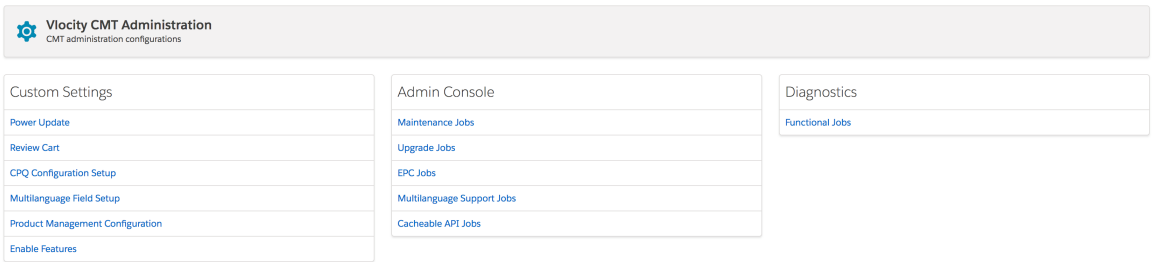
You can configure the catalog to support over 25 different languages. Before applying these languages, you must configure the multi-language support feature for your shared catalog:

1. Click **Setup**.
2. In the **Quick Find / Search** field, type **`Picklist Value Sets`**.
3. Click **Vlocity Locale Code**.
   The Global Value Set (Managed) page appears.



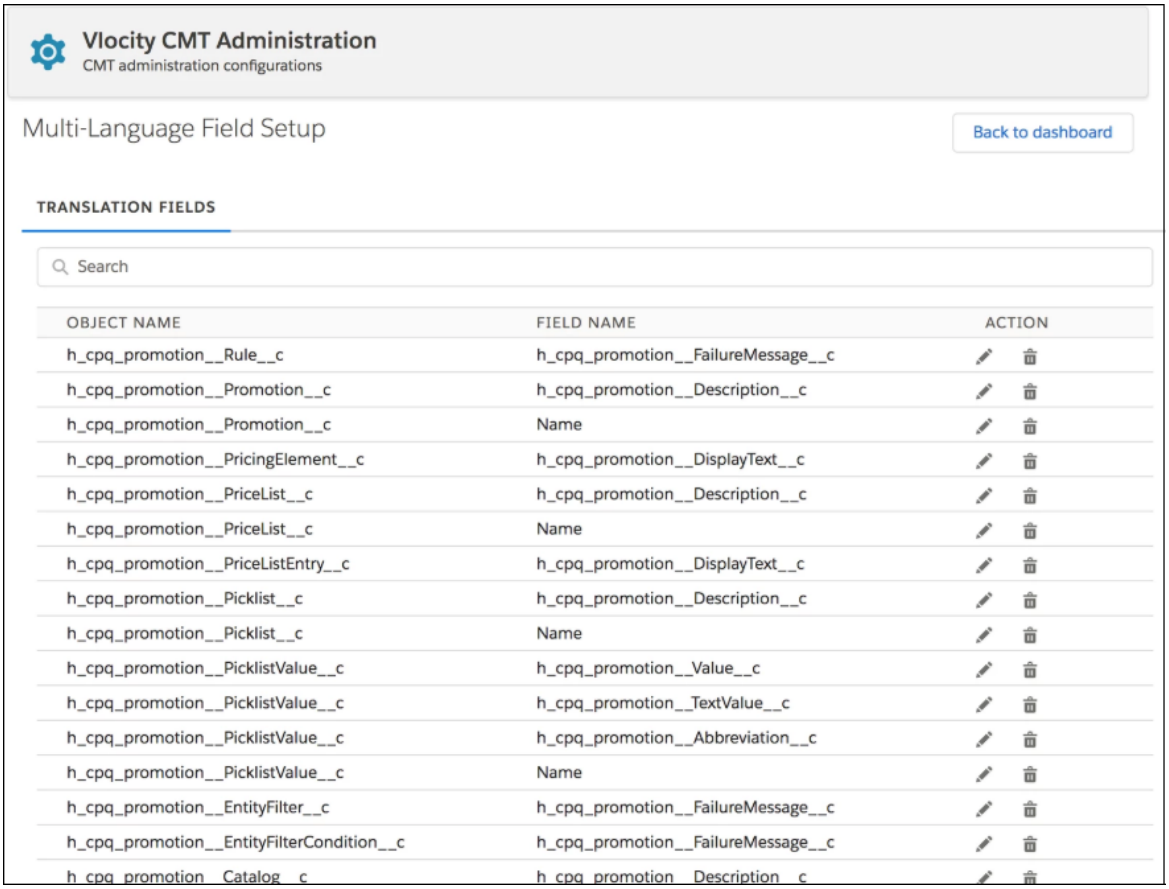
4. Check that an activated language is set as the default, such as **English**.
5. Activate or deactivate values based on required locale codes for translations. Activate the required locale code and deactivate all of the other codes. Set the API Name to the Salesforce Locale code.

6. Enable Multi-Language Support for the Shared Catalog so that product data and values that have translations are translated before appearing in the UI. This setting allows product administrators to translate product data, and also allows customer service representatives (CSRs) to view translated data.
   a. Go to **Vlocity CMT Administration**.



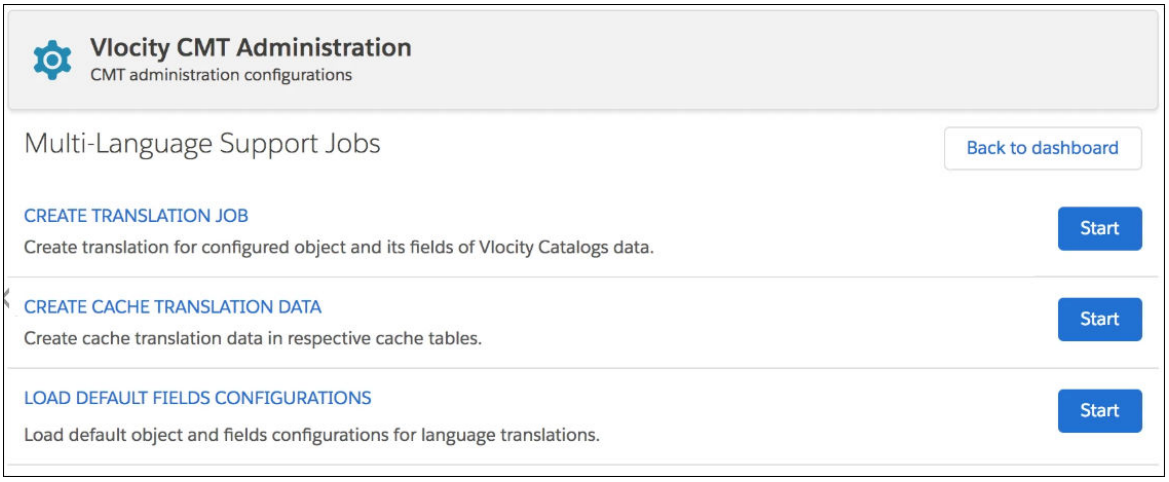
   b. Click **Enable Features**.

   c. For Multi-Language Catalog Support, click **Enable**.
   If the feature is disabled, no translations occur and product data and values are displayed as entered.
7. Configure objects and their fields for multi-language support. There's a fixed set of fields for products, promotions, attributes, and picklist configured for translation. For example, if a customer extends the Product2 object to add custom fields, then those custom fields that require translation must be configured as follows:
   a. Go to Vlocity CMT Administration.
   b. Under Custom Settings, click **Multi-Language Field Setup**.



c. Perform any of the following operations:
   - Click **Add** at the bottom of the page to add a record.
   - Click the **Edit** icon to modify a record.
   - Click the **Delete** icon to delete a record.

8. Process the objects configured in the previous step:
   a. Go to Vlocity CMT Administration.
   b. Click **Multi-Language Support Jobs**. The Multi-Language Support Jobs page appears.



c. Click **LOAD DEFAULT FIELDS CONFIGURATIONS**. This job loads out of box objects and fields configuration for multi-language support. But it doesn't remove any data added by users in step 6. This job must be run by the product admin whenever there's a change to Multi-Language Field Setup.
The following table lists the out of box data:

Object Name	Fields	Description
Attribute__c	Value__c	Default value of an attribute.
Attribute__c	HelpText__c	Help text to display more information about the attribute.
Attribute__c	Description__c	Short description of an attribute to help identify the purpose of the attribute.
Attribute__c	Name	Name of an attribute. Names are used to identify attributes in search.
AttributeAssignment__c	Value__c	Default value of an attribute assigned to an Object Type.
AttributeAssignment__c	ValueDescription__c	Display name of the attribute assigned to an Object Type.
AttributeAssignment__c	ValidValuesData__c	This field stores related data into JSON format when you assign picklist or multipicklist attributes to a product from the Product page in Aloha.
AttributeCategory__c	Name	Name of the Attribute Group. Attributes are displayed at runtime by attribute group name.
Catalog__c	Description__c	Short description of the Product Catalog. Description helps to identify the purpose of the Product Catalog.
Catalog__c	Name	Name of the product catalog.
CatalogRelationship__c	Name	Name of the nested product catalog.
EntityFilter__c	FailureMessage__c	Failure message displayed at runtime by Entity Filter.
EntityFilterCondition__c	FailureMessage__c	Failure message displayed at runtime by Entity Filter.
Picklist__c	Description__c	Short description of the Picklist. Description helps to identify the purpose of the Picklist.
Picklist__c	Name	Name of the Picklist.
PicklistValue__c	Value__c	Code of the Picklist Value.
PicklistValue__c	TextValue__c	Value of the Picklist.

Object Name	Fields	Description
PicklistValue__c	Abbreviation__c	Abbreviated name of the Picklist value.
PicklistValue__c	Name	Name of the Picklist value.
PriceList__c	Description__c	Short description of the Pricelist.
PriceList__c	Name	Name of the Pricelist.
PriceListEntry__c	DisplayText__c	Display text of the Pricelist Entry at runtime.
PriceListEntry__c	DisplayText__c	Display text of the Pricelist Element at runtime
Product2	Description	Short description of the Product.
Product2	Name	Name of the Product.
Promotion__c	Description__c	Short description of the Promotion.
Promotion__c	Name	Name of the Promotion.
Rule__c	FailureMessage__c	Failure message displayed at runtime by Rules.

d. Click **CREATE TRANSLATION JOB**. This job goes through all the configured objects and their fields, collects strings, and populates the StringTranslation Object for translation.
Before starting this job, make sure you have the correct configuration for objects and fields as described in Multi-language Field Setup.

9. From the Multi-Language Support Jobs page, click **CREATE CACHE TRANSLATION DATA**. This job creates cache data for products and promotions to support search and sort in getProductList and getPromotionList. This job copies data from StringTranslation__c and Product/Promotion objects and fills CachedProduct2Translation__c and CachedPromotionTranslation__c.
The following table is for CachedProduct2Translation__c, which stores translation data for a product:

Field API Name	Datatype	Description
LocaleCode__c	Picklist	Possible locale list
Product2Id__c	Lookup(Product2)	Reference to Product2
Product2Name__c	Long TextArea	Translated product name
Product2Description__c	TextArea	Translated product description

The following table is for CachedPromotionTranslation__c, which stores translation data for a promotion:

Field API Name	Datatype	Description
LocaleCode__c	Picklist	Possible locale list
PromotionId__c	Lookup(Product2)	Reference to Promotion__c
PromotionName__c	Text	Translated promotion name
PromotionDescription__c	TextArea	Translated promotion description

10. Create an Opportunity or Order.
The translated values appear in the CPQ cart, Review Cart Visualforce page (CardFramework), and Assets VF page (CardFramework).
If a string translation is missing for labels in the selected view from Order Cart, ensure the names of the Card States follow the convention CustomViews_<<label>>. Then, the CPQ service translates the display for each custom label value found in that corresponding org. The labels display if the translations associated with the language are present in the system.

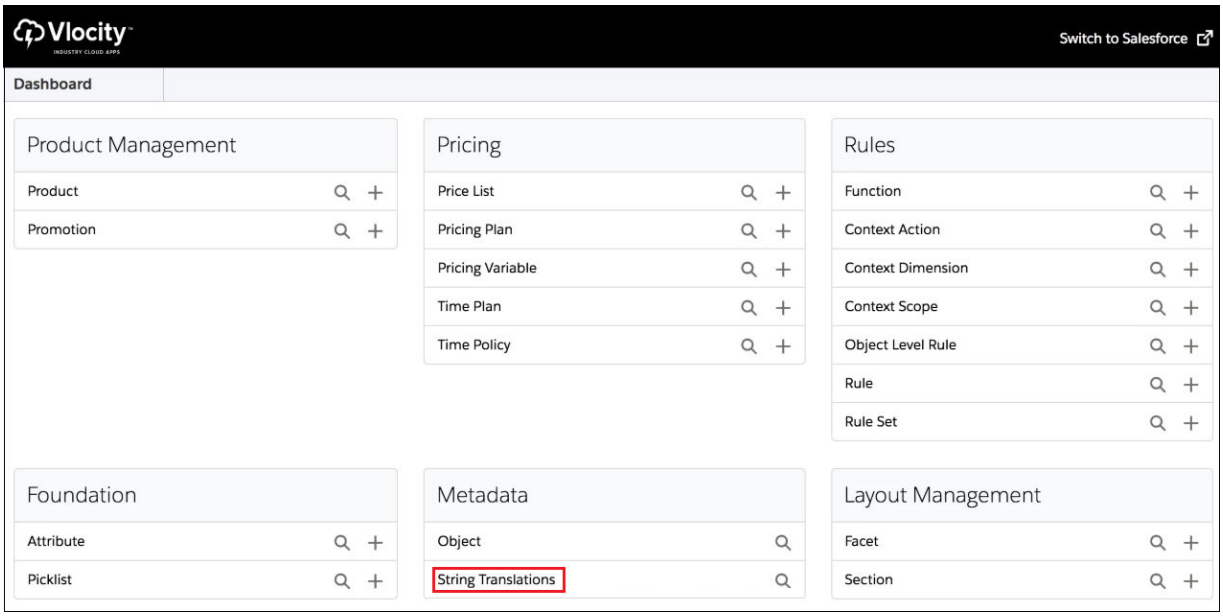
Modify String Translations in Vlocity Product Console

Product administrators who are responsible for product catalog administration and life cycle or contract generation configure the base language and can modify the translated strings in the Vlocity Product Console.

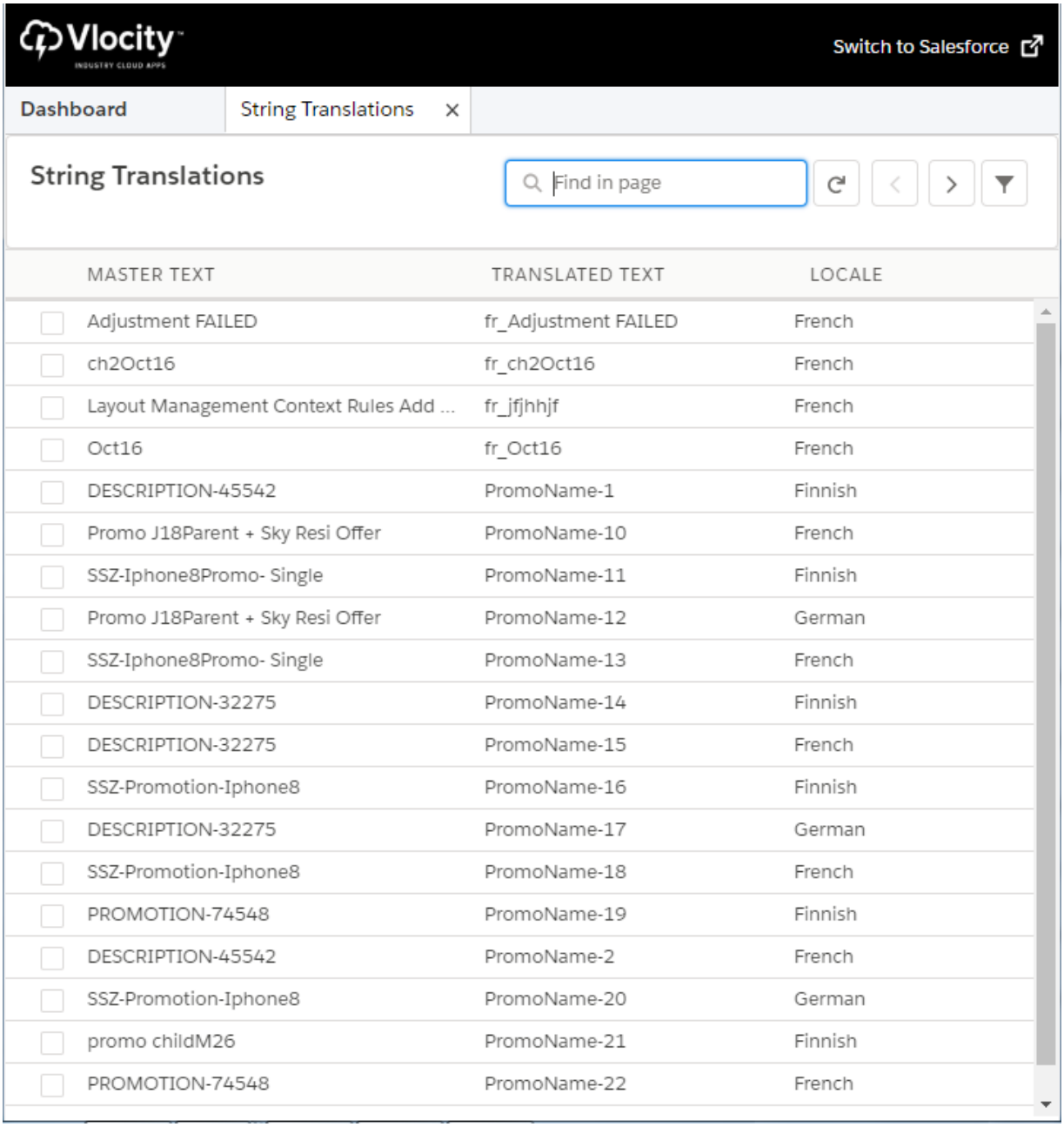
Before You Begin
1. Activate the needed locales (languages). 2. Enable multi-language support in Vlocity CMT Administration. 3. Configure objects and their fields for multi-language support. 4. Run the Create Translation job to extract product data (base language) for translation. For details on these tasks, see Configuring Multi-Language Support for Shared Catalog

To modify translated strings:

1. Go to the Vlocity Product Console.



2. Click **String Translations**.



From this page, you can:

- Search for master text and translated text
- Click the filter icons to filter by master text, translated text, and locale.
- Click the column headings to sort by master text, translated text, and locale.
- Click the Edit icon for any row to edit the associated translation text.

Ensure that the Locale field matches the Values field from the Global Value set for Vlocity Locale Code.

3. If editing a string, type the translation into the **Translated Text** field, and click **Save**.

See Also

- Multi-language Support for Shared Catalog
- Configuring Multi-Language Support for Shared Catalog
- Set String Translations in Product Designer

Set String Translations in Product Designer

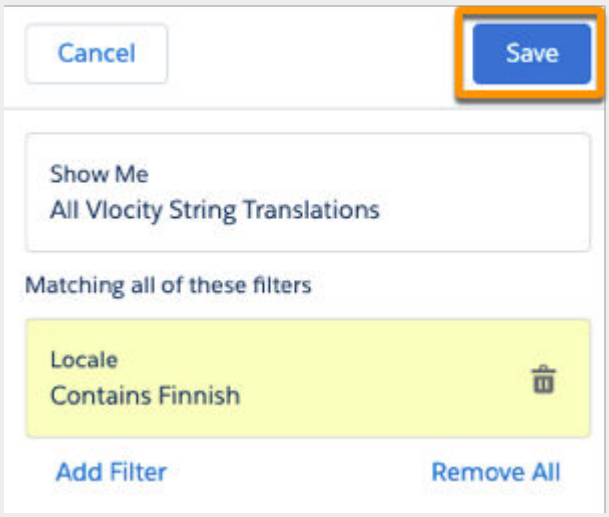
Create translated text for names and descriptions for catalog entities, such as attributes, picklists, and products. The translated text is shown in the cart for the appropriate locale. You can modify the translated strings in the Vlocity Product Designer.

Product administrators who are responsible for product catalog administration and lifecycle or contract generation configure the base language.

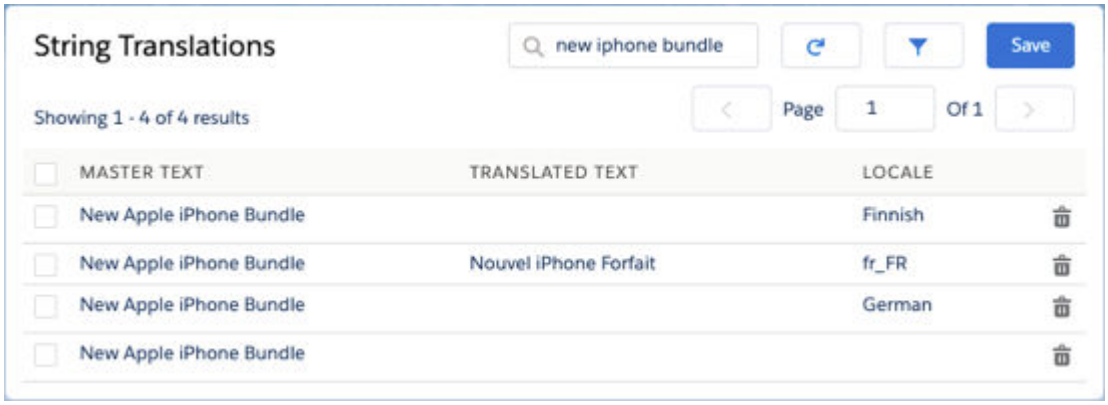
Before You Begin
1. Activate the needed locales (languages). 2. Enable multi-language support in Vlocity CMT Administration. 3. Configure objects and their fields for multi-language support. 4. Run the Create Translation job to extract product data (base language) for translation. For details on these tasks, see Configuring Multi-Language Support for Shared Catalog.

1. From the Vlocity Product Designer list, click **Vlocity String Translation**.
2. Find catalog entities to enter or edit the translated text.

Goal	Action
Search master or translated text.	Enter text

Goal	Action
Filter the catalog entities by master text, translated text, or locale values.	1. Click the **Filter** button. 2. In the Filter panel, click **Add Filter**. 3. In the New Filter dialog box, set the Field, Operator, and Value for the filter, and click **Done**. 4. In the Filter panel, click **Save**.  The list shows only entities that match the filter.
Sort entities by master text, translated text, or locale values.	Click a column heading to sort values in ascending or descending order.

3. In the Translated Text column, click the pencil icon to enter the translated value for a catalog entity, and click outside the field to temporarily save the edit.
   The translated text isn't saved.



4. Click **Save**.
5. Repeat steps 2–4 for catalog entities and locales as needed.
6. From the Multilanguage Support Jobs page in Vlocity CMT Administration, click CREATE CACHE TRANSLATION DATA.

This job creates cache data for products and promotions to support search and sort in getProductList and getPromotionList. This job copies data from StringTranslation__c and Product/Promotion objects and fills CachedProduct2Translation__c and CachedPromotionTranslation__c.

See Also

- Multi-language Support for Shared Catalog
- Configuring Multi-Language Support for Shared Catalog
- Modify String Translations in Vlocity Product Console

Translating Freeform Text Attributes

As an administrator, you can use this feature to configure freeform text attributes as either translatable or not-translatable. This enables you to specify what product attributes are translated into other languages. In previous releases, all product attributes (string data-type attributes) were translated, regardless of whether they were freeform or picklist-enabled. As of Winter '19 release, you can configure freeform string attributes to specify if the string is translatable. For example, you might not want to translate email addresses, locations, or comments in freeform string attributes.

To configure the text attributes as either translatable or not-translatable:

1. From App Launcher, click Vlocity Objects and Object Types.
2. Create or update an Object Type definition.
3. Assign an existing text attribute. Do not use a picklist.
4. Uncheck the flag to indicate the attribute is not translatable.
5. Ensure all the translation activities are performed for other attributes and fields. For more information, see the other topics in Multi-language Support for Shared Catalog.
6. Log into your org with a non-English locale, such as French or Spanish.
7. Create an order with a product that contains the translatable attribute.
8. Enter values for the attribute in, for example, English and click **Save**.
9. Reopen the order and verify the attribute value. It should be in English, the original language in which the attribute value was entered.

Multi-Language Support for All Catalog Elements in CPQ

Multi-language support for Vlocity Product Catalog enables CPQ to render information in various languages. Within CPQ, the following are translated:

- All out of the box labels and field text
- All custom labels and field text
- Review page and Asset Cart
- Attributes and attribute values generated during CPQ configuration
- System-generated error messages from Vlocity
- Custom-generated messages, including error messages defined as part of Rules

You can use the translated language to search for and sort existing features within CPQ, including promotions and products.

When generating a document using the CLM document generator, CLM can retrieve and translate header and xLI information.

Migrating Translation Data Across Salesforce Orgs

CPQ can identify Vlocity and Salesforce objects and their fields used by the Shared Catalog (EPC) for values that can be translated using the UI.

The Vlocity Build tool enables you to migrate product data with translations from one environment to another, for example, from a development environment to a test environment, assuming standardized environments and the same version. The Vlocity Build tool also enables you to migrate product data with translations from older versions of Salesforce+Vlocity CME environments to new versions of Salesforce +Vlocity CME environments.

There is also support for Product API methods (GET) to query by a specific language, and support for validation of product data translations for completeness and consistency, for example, an export of all picklist data. You can use data packs to transfer String data and String Translation data from one org to another.

Customers can use Data Raptor (field implementation) to import translations from legacy systems.

Translating the Vlocity Cart UI

You can translate the field labels, headers, and other UI elements of the Vlocity Cart using a custom language file.

To translate the cart UI:

1. Enable a translation file:
   a. Log in to Salesforce.
   b. Go to **Setup**.
   c. In the Quick Find box, type **`Translation Workbench`**.
   d. From the search results, click **Translation Settings**.
      The Translation Settings page opens.
   e. Click **Enable**, if it is not already selected.
2. Add a new (supported) language. See Vlocity Locale Codes.
   a. On the Translation Settings page, click **Add**.
      The New Language page opens.
   b. From the **Language** drop-down list, select the language you wish to add, for example, English.
   c. To assign a user as a translator for the selected language, select the user in the **Available List** , and add the user to the **Selected List**.
   d. Ensure that the **Active** field is checked.
   e. Click **Save**.
3. Export the language file:
   a. Go to **Setup**.
   b. In the Quick Find box, type **`Translation Workbench`**.

c. From the search results, click **Export**.
The Export page opens.
d. Select **Untranslated** and click **Export as STF**.
You will be notified by an email when the export is complete.
e. Follow the link in the email.
f. In the **Recent Documents** section, click the document named `Untranslated_timestamp.zip`.
The Document Details page opens.
g. Click **View file** to download the file.
The downloaded file contains a list of resource ids and the corresponding labels in the selected language, in this case, English.
h. Unzip the `.zip` file.

4. Create translations:
You can create translations by entering a command or by using a Python script.
a. In your terminal, navigate to the folder containing the downloaded `.stf` file.
Use **sed $'s/\t/&**/g' source.stf > output.stf** command.
This command appends ** before each string value.
Example: **sed $'s/\t/&**/g' Untranslated_iw_2018-03-09\ 0112.stf > Translated.stf**
Open the `Translated.stf` file and verify that there are ** before each label.

NOTE
Before importing the `Translated.stf` file, open the file and shorten the field labels if they are longer than 40 characters, for example, 'OverriddenAttributeAssignmentCategoryId', 'OverriddenOfferPricingComponentId'. Remove some characters from the label names, if not the import will fail due to 40 characters limitation. If the labels are already 40 characters long, adding ** will make their length to be 42 characters causing the import to fail.

b. Alternatively, you can use the following Python script to create translations:

```
----------------------------------------
import sys

# Take two files as input (the last one might not exist if it does
exist the content is appended.
#The first file represents the input file with un-translated strings.
#The second one represents the output file where the translations will
be written.
##
#The translation process consists of taking strings of the form:
# ButtonOrLink.vlocity_cmt__Application__c.vlocity_cmt__Review    Review
#and converting them to:
# ButtonOrLink.vlocity_cmt__Application__c.vlocity_cmt__Review
```

```
**Review
#
# by adding "**" after the tab.
#
#In the case that after the translation a string is longer than 40
characters (and its key contains '.FieldLabel')
# then the last characters are removed.

def main():

    input = sys.argv[1]
    output = sys.argv[2]
 with open(input, 'r') as input_file:

        for line in input_file:

            line = line.strip()
            if '\t' in line:
                line = line.replace("\t", "\t**")

                line = removeExtraCharacters(line)

            with open(output, 'a') as output_file:
                output_file.write(line + '\n')
def removeExtraCharacters(input):

    if '.FieldLabel' not in input:
        return input

    keyValuePair = input.split('\t')
    key = keyValuePair[0]
    value = keyValuePair[1]

    length = len(value)
    if length <= 40:
        return input
    else:
        value = value[:40]
        output = key + '\t' + value
        return output


if __name__ == '__main__':
    main()
-------------------------------------------------
```

5. Import the translated file:

a. In the Quick Find box, type **`Translation Workbench.`**
b. From the search results, click **Import**.
   The Import page opens.
c. Click **Choose File**.
d. Select the file you edited for translations.
e. Click **Import**.

6. Change the language in your org:
   a. In the **Search Setup** box, type in your user account name.
      The User page opens.
   b. Click **Edit**.
      the User Edit page opens.
   c. Scroll to the **Locale Settings** section and set the **Locale** and **Language settings**.
   d. Click **Save**.
      Now, most labels on your org render in the selected language. However, note that you cannot translate Tabs, Objects, and Record names. The label for the 'name' field in every Salesforce object must be translated manually as Salesforce needs grammatical information.

To translate labels manually:

1. Go to the Quick Find box and type **`Rename Tab and Labels`**.
2. Select Rename Tabs and Labels from the search results.
   The Rename Tabs and Labels page opens.
3. From the **Select Language** drop-down list, select the language you want to rename the labels into.
   The labels are translated and are displayed in the **Display label** column.

Vlocity Locale Codes

This table lists the Vlocity locale codes that you can activate for translating label names. This is an out-of-the-box list that Vlocity provides. You can activate only three locale values at a time. See Activating Vlocity Locale Code Values

Table 7. Vlocity Locale Codes

Language	Local Code (API name)	IsActive
Finnish	fi	Yes
French	fr	Yes
German	de	Yes
Chinese	zh_CN	No
Chinese (Traditional)	zh_TW	No
Danish	da	No
Dutch	nl_NL	No
English	en_US	No
Italian	it	No
Korean	ko	No

Language	Local Code (API name)	IsActive
Norwegian	no	No
Portuguese (Brazil)	pt_BR	No
Russian	ru	No
Spanish	es	No
Spanish (Mexico)	es_MX	No
Swedish	sv	No
Thai	th	No