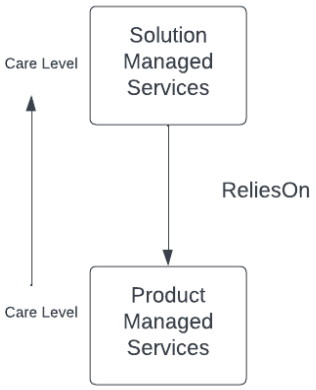
Propagate/Copy Attributes via ReliesOn Relationship

Author: Jeevanprakash Chinya Manjunatha
Technical Architect, Salesforce
Date: 15/June/2023

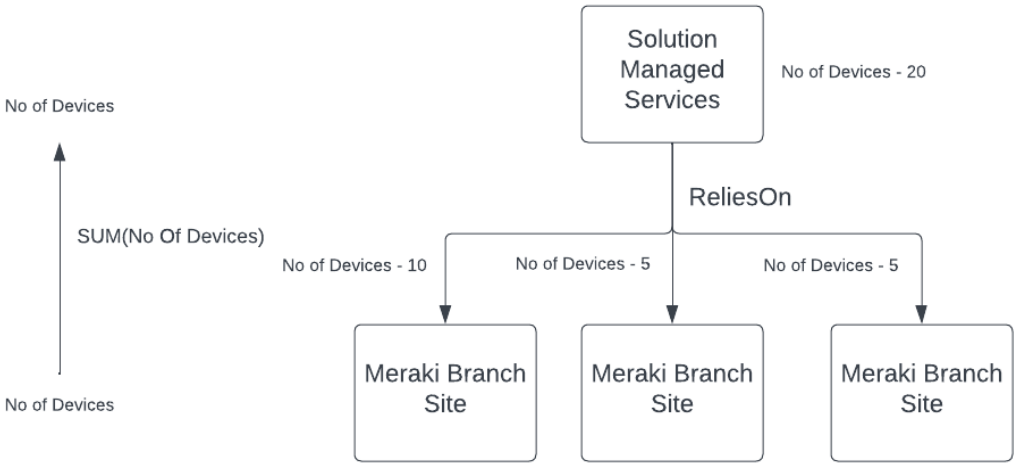
Problem Statement

When products are related via ReliesOn relationships, customers want a capability to propagate attributes from related product instances to the source product instance.
This will help users to not configure the same attributes during cart configuration in the multiple product instances and such attributes might be required to send for downstream systems.

Ex: In below example, Solution Managed Services relieson Product Managed Services and systems needs to propagate Care Level attribute from Product Managed Services to Solution Managed Services.



In below example, Solution Managed Services relieson Meraki Branch Site and systems needs to propagate SUM of No of Devices from all instances.



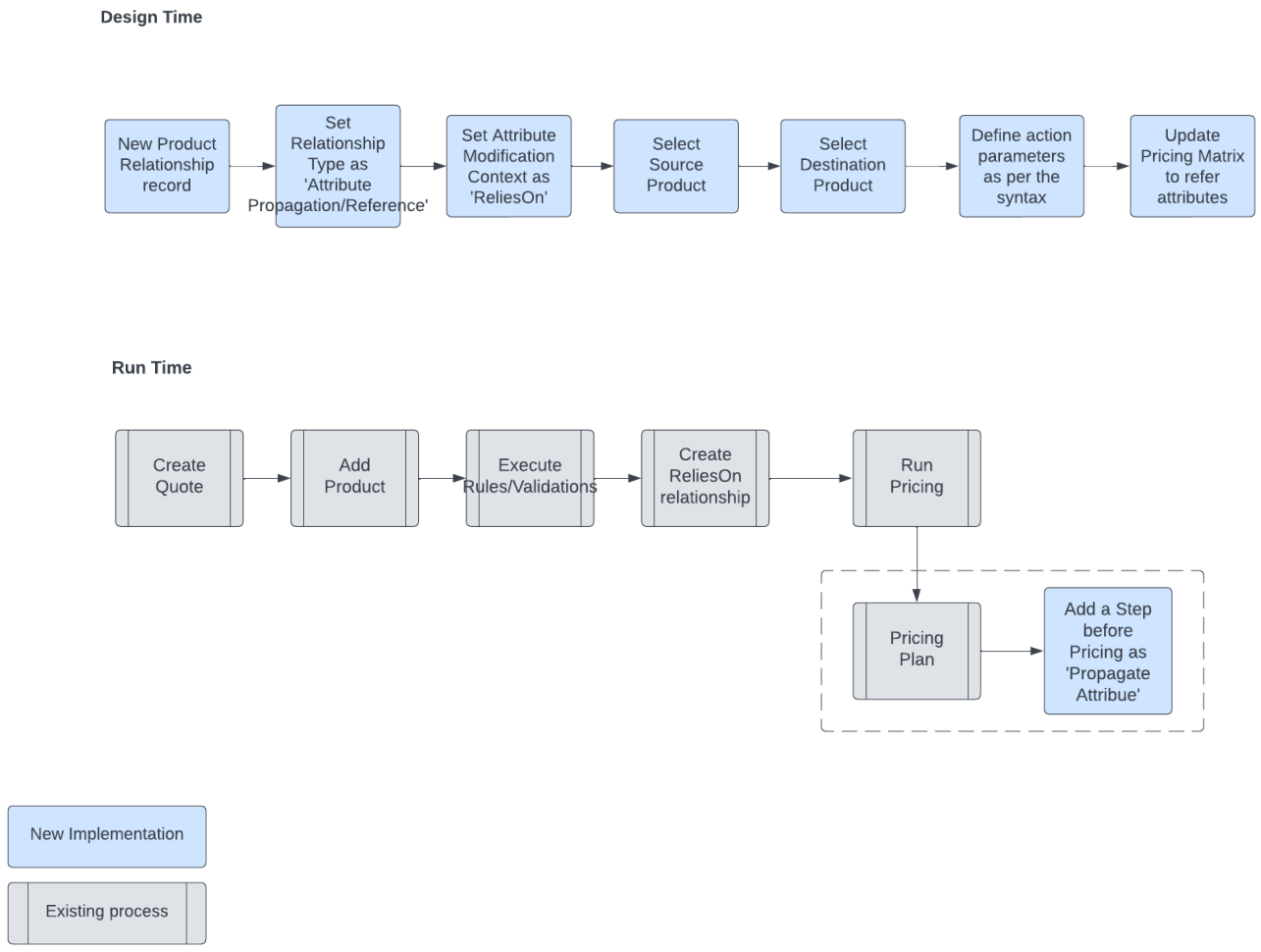
Generally this kind of requirement is achieved using hook implementation but this will be a static solution which needs change or development effort for every new product or change in product.

Technical Design

This feature design will be split into two phases, design time configuration and run time engine.

Design time configuration, which provides ability for product admin to define attributes which need to be propagated.

Run time engine, which refers to the design time definition and current cart product instances and propagate attributes.



Design Time

Extend SFI OOTB, Product Relationship object(vlocity_cmt__ProductRelationship__c).

1. Add a new Picklist value for picklist Relationship Type field as ‘Attribute Propagation’
2. Add a new Picklist value for picklist Attribute Modification Context as ‘ReliesOn’

Use the ‘Action Parameter’ field to capture attributes that need to be captured from Related Product Instance to the Source Product Instance.

As part of this example we are supporting direct attribute propagation and also SUM of attribute instance propagation (if relationship is 1:M and depends on attribute data type).

> 💡 **Tip**
> **In this example, attribute propagation is done for ReliesOn context only. This can be extended if attributes need to be propagated within the bundle, product hierarchy or cart context. Attribute Modification Context field has OOTB options such as Parent, Bundle and Cart. This could be used as an identifier in the implementation class to fetch attributes from related products. Do an impact analysis to make sure this will not coincide with OOTB implementations.**

Syntax

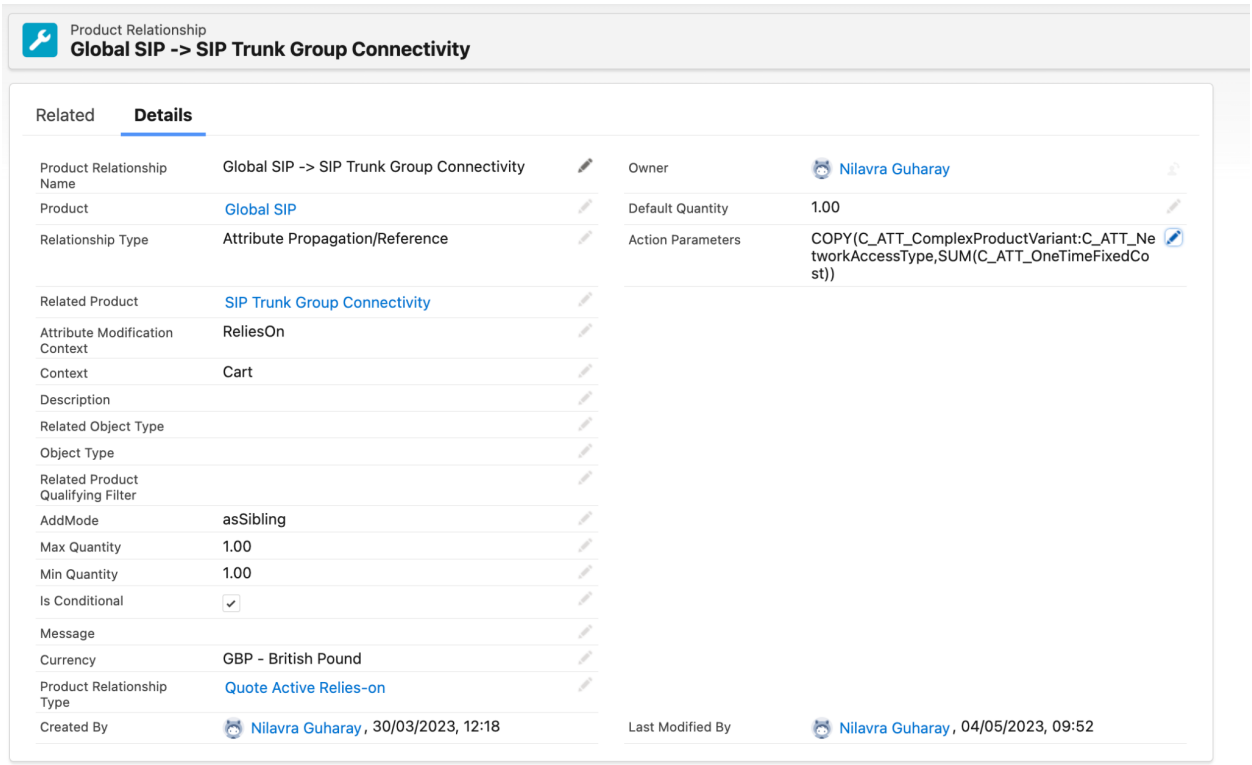
Copy(AttributeCode, SourceAttributeCode:RelatedAttributeCode , SUM(AttributeCode))

AttributeCode - If Source and Related product has same product code which needs to be copied

SourceAttributeCode:RelatedAttributeCode - RelatedAttributeCode from Related product will be copied to SourceAttributeCode of Source product

SUM(AttributeCode) : Attribute code rom Related products will be summed up to AttributeCode of Source product

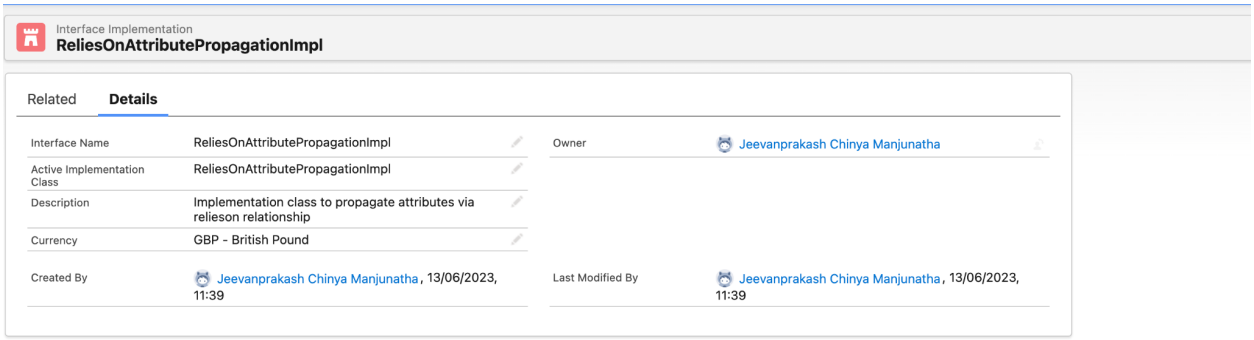
> 💡 **Tip**
> **In this example, we have done only attribute propagation. Even this can be extended to propagate field values as well. Define unique syntax to differentiate attributes vs fields.**



Run Time Engine

Create a new Interface implementation called ***'ReliesOnAttributePropagationImpl'*** which parse created Relies On relationship instances and propagates attribute values based on relies on attribute propagation definition and SUM of same attribute to be propagated based on the relies on attribute propagation definition.

Interface checks for the scope of the invocation either 'Quote' or 'Order', and check respective relationship object for the relies on relationship instance (Quote: vlocity_cmt__QuoteLineItemRelationship__c and Order: vlocity_cmt__OrderItemRelationship__c)



Interface Implementation can be an invocable class, by passing below I/P.

Inputs:

Scenario 1: Invoked in pricing plan step

invokedFromPricing = true (Required)

Pricing context already has details of lineitems for which pricing is being executed. So no need to pass Id's explicitly.

Scenario 2: Invoked by custom process

invokedFromPricing = false (Required)

ContextId (For Quote its Quote.id and for Order it should be Order.id). When context id is passed, attribute propagation will execute for all the relies on relationship instance exist for that Quote/Order (Bulk execution)

RecordIds (For Quote its list of Quote Line Item Ids and for Order it should be list of Order Item Ids). When RecordIds is passed, attribute propagation will execute only for the relies on relationship instance exist for that lineitem list

Either ContextId or RecordIDs is required if invokedFromPricing is false.

At a time, we need to pass either ContextId or RecordIds as input. Default value for invokedFromPricing is false.

Output:

The class would propagate the attributes from the target to the source and would return the scope (Quote or Order) and the list of Line Item Ids (Order Items or Quote Line Items) for which the attributes were copied to and send back the output in this format


```
{
  "totalSize": 1,
  "messages": [
    {
      "message": "Attributes are updated successfully"
```

```
    }
  ],
  "records": [
    {
      "Scope": "Quote",
      "Records": idslist
    }
  ]
}
```

> 📒 **Note**
> **Parsing attributes from JSON might impact execution time depending on the number of attributes and size of the Quote or Order. It's better to invoke this implementation as an Asynchronous job to handle large Quotes or Orders. In case of invocation in pricing plan step, do an analysis of the Quote/Order size vs execution time and handle pricing accordingly as an batchable process so only set of QLI's/OLI's will be available in the pricing context per batch.**

Performance Metrics

Metrics	APEX Execution Time	APEX Execution Time
No of Line Items in Quote: 1 **No of Product Relationship Definition: 1** **Average no of Attributes per line item: 2** **Quote Product Relationships - 1**	670 ms	656 ms
No of Line Items in Quote: 100 **No of Product Relationship Definition: 3** **Average no of Attributes per line item: 5** **Quote Product Relationships - 52**	919 ms	923 ms

No of Line Items in Quote: 1000 **No of Product Relationship Definition: 3** **Average no of Attributes per line item: 5** **Quote Product Relationships - 800**	2953 ms	3015 ms

> 📒 **Note**
> **This feature is tested in 242 CMT versions and it is expected to work in higher versions of CMT packages as well.**

Credits

Design & Author	Jeevanprakash Chinya Manjunatha
Solution Architect & Reviewer	Francisco Pina
Developers	Jeevanprakash Chinya Manjunatha Nilavra Guharay