The SuperOffice Quote Connector Interface

**Change history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Who** | **Comments** | **Version** |
| 2012-12-20 | TFN | First official version | 0.7 |
| 2013-01-17 | TFN | Second version | 0.8 |
| 2013.01.24 | TFN | Updates from workshop | 0.81 |
| 2013.03.12 | TFN | Updates from sprint | 0.9 |
| 2013.05.23 | TFN | Updates from meeting | 0.91 |
| 2013.06.04 | TFN | Updates to OrderConsumer | 0.92 |
| 2013.06.19 | TFN | Removed Interfaces (This is handled by capabilities)  Inserted response objects | 0.93 |
| 2013.09.03 | CM | UpdatePrices, Validation changes | 0.94 |

We expect next update to be when we are feature complete.

Table of Contents

[Background & Vision 1](#_Toc365989338)

[Architecture 1](#_Toc365989339)

[Files 1](#_Toc365989340)

[Quote the SuperOffice way… 1](#_Toc365989341)

[Development 1](#_Toc365989342)

[The SuperOffice Quote Management API 1](#_Toc365989343)

[Some facts 1](#_Toc365989344)

[Company Policy Preferences 1](#_Toc365989345)

[Other company policy rules 1](#_Toc365989346)

[Parts 1](#_Toc365989347)

[Quote Connector Setup #9537 1](#_Toc365989348)

[Basic Connector #9538 1](#_Toc365989349)

[Capability Names 1](#_Toc365989350)

[Product Provider #9539 1](#_Toc365989351)

[Product Search Provider 1](#_Toc365989352)

[Quote List Provider #9540 1](#_Toc365989353)

[Quote List Names 1](#_Toc365989354)

[Price Provider #9541 1](#_Toc365989355)

[Sending Quotes 1](#_Toc365989356)

[Order Consumer #9542 1](#_Toc365989357)

[Address Provider 1](#_Toc365989358)

[IArchiveProvider 1](#_Toc365989359)

[IProductRegisterCache #9543 1](#_Toc365989360)

[Data carriers 1](#_Toc365989361)

[QuoteConnectionInfo #9544 1](#_Toc365989362)

[ISaleInfo 1](#_Toc365989363)

[QuoteInfo 1](#_Toc365989364)

[QuoteVersionInfo 1](#_Toc365989365)

[QuoteVersionStateInfo 1](#_Toc365989366)

[QuoteAlternativeInfo 1](#_Toc365989367)

[Quote Alternative Discounts/Earning 1](#_Toc365989368)

[Cost + Earning = SubTotal – Discount = TotalPrice 1](#_Toc365989369)

[QuoteLineInfo #9546 1](#_Toc365989370)

[Enum ValueOverrideInfo 1](#_Toc365989371)

[Rights field 1](#_Toc365989372)

[QuoteListItemInfo 1](#_Toc365989373)

[QuoteVersionResponseInfo 1](#_Toc365989374)

[QuoteSentResponseInfo 1](#_Toc365989375)

[OrderResponseInfo 1](#_Toc365989376)

[PlaceOrderResponseInfo 1](#_Toc365989377)

[FieldMetadataInfo 1](#_Toc365989378)

[Enum ConfigFieldType 1](#_Toc365989379)

[Enum FieldAccessInfo 1](#_Toc365989380)

[Config values 1](#_Toc365989381)

[PluginResponseInfo 1](#_Toc365989382)

[Enum QuoteStatus 1](#_Toc365989383)

[Error system 1](#_Toc365989384)

[ERP Discounts and User Discounts 1](#_Toc365989385)

[Value Fields and Amounts and Percent’s 1](#_Toc365989386)

[QuoteAlternativeWithLinesInfo 1](#_Toc365989387)

[QuoteAlternativeContextInfo #9545 1](#_Toc365989388)

[QuoteVersionContextInfo 1](#_Toc365989389)

[PriceListInfo #9547 1](#_Toc365989390)

[ProductInfo (Article) #9548 1](#_Toc365989391)

[ProductExtraDataFieldInfo 1](#_Toc365989392)

[ExtraDataFieldTypeInfo 1](#_Toc365989393)

[QuoteConnectorExtender 1](#_Toc365989394)

[AddressInfo 1](#_Toc365989395)

[Use Cases 1](#_Toc365989396)

[Use Case – Adding a Product 1](#_Toc365989397)

[Use case – Dealing with customer specific product codes 1](#_Toc365989398)

[QuoteConnectorBase implementation 1](#_Toc365989399)

[QuoteConnectorExtender implementation 1](#_Toc365989400)

[How to create a SuperOffice Quote Connector 1](#_Toc365989401)

[Prerequisites 1](#_Toc365989402)

[Basics 1](#_Toc365989403)

[If you just want to replace or extend part of an ERP connector 1](#_Toc365989404)

[If you don’t want to extend an existing ERP connector 1](#_Toc365989405)

[When Done 1](#_Toc365989406)

# Background & Vision

The SuperOffice Quote Management system is based on an architecture that allows connection to several ERP systems. This is facilitated through a set of “Quote Connectors”. A quote connector provides specific data- and business logic for the ERP system in question. A Quote Connector communicates with SuperOffice through a set of API’s which are specifically made available for connector purposes. SuperOffice relies on partners to develop all Connectors. SuperOffice will certify all available Connectors, but will not distribute them or offer them as part of our standard pricelist.

## Architecture

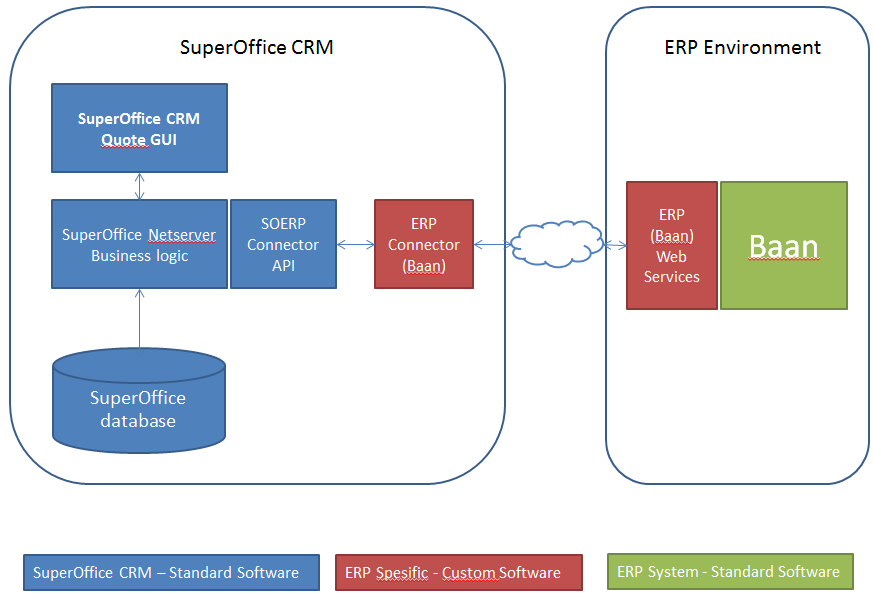


Figure 1: ERP system in this case is BaaN

The <**SpesificERP>QuoteConnector.DLL** is loaded into the SuperOffice client when the SuperOffice client starts.

The information needed to connect to the ERP system is set up and stored in the SuperOffice database first.

## Files

**SoDatabase.dll** – Contains the business logic and the implementation of the core functionality. It changes with every major and minor release of SuperOffice.

**SuperOffice.Plugins.Dll** – Contains the interface definitions and data carriers used by the interface. It changes rarely, and then only additions. The goal is that ERP plugins only need to reference the plugins.dll, so that they are compatible across minor and major releases of SuperOffice.

**Erp.QuoteConnector.Baan.dll** – Contains the implementation of the IQuoteConnector interface defined in SuperOffice.Plugins.dll. This DLL should be compatible across minor releases of SuperOffice as long as it does not reference the SoDatabase.dll directly.

## Quote the SuperOffice way…

We shall try to make the API as easy to use as possible, but without sacrificing the simple and user friendly user interface we want to make.

## Development

We will use develop this system in an agile manner, that means that most of the way will be found as we go. This means that some of the features we discuss here might not be developed in time for the first release.

# The SuperOffice Quote Management API

The API will be implemented as an interface named IQuoteConnector, of which an ERPConnector can implement one or more functions. By overloading either BaseQuoteConnector or QuoteConnectorExtender you get to only overload the parts you want to change.

Functions that we expect to vary between ERP systems we will make queryable via capability checks. e.g.: Not all connectors will support the creation of orders, so SuperOffice will first check that the capability is available by calling CanProvideCapability(“iorderconsumer\_place\_order”)

# Some facts

* A SuperOffice installation can have, 0, 1 or many ERP connectors at the same time. Many large companies have more than one ERP system. (Typically divided over country borders.)
* The connector should be totally without user interface. It might be run at a server far, far away, far away from the user, by both Windows and Web clients.
* A connector must be installed and configured by the administrator.
* The system will allow the administrator to set up which salesmen shall have access to which ERP clients. If a salesman has access to more than one system, he will be asked which one he wants to use when he creates a quote.
* Since not all connectors will be able to support all functionality, and we don’t want to accept the least common denominator, the connector should be query-able; SuperOffice shall be able to query the connector about its capabilities.
* SuperOffice shall report to the connectors the language the user is running in, and will strongly request that the responses is translated as far as possible (especially the user error responses).

## Company Policy Preferences

SuperOffice Quote system can be configured to enable the various features available:

* Alternatives – available or not.
* Versioning – used or not.
* Discounts on total order amount – or only on line items.

The preferences and their effects are described in the main QUOTE MANAGEMENT spec.

# Other company policy rules

See QuoteConnectorExtender chapter for a simple way of adding a specific rule for an installation without having to recompile the whole erp connector.

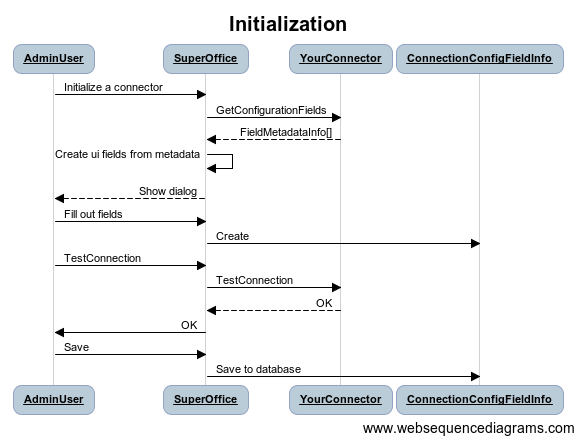
# Parts

The system is implemented as one interface, but has some parts:

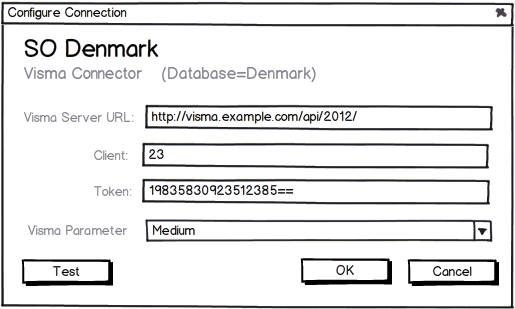
|  |  |
| --- | --- |
| Quote Connector Setup | Configure the connection to the ERP system. Provides meta-data about connection properties. |
| Basic Connector | Handles the connection to the ERP system and the work of keeping the ERP system informed about the Quote. |
| Product Provider | Search products and price lists. |
| Price Provider | Calculates the discount on a single quoteline, and on a complete quote. |
| Order Consumer | Accepts a quote and creates an order in the ERP system. |
| Address Provider | Some ERP systems will be able to supply the default addresses for a quote/order. |
| Product Search Provider | A set of functions to be able to do more complex search. |

## Quote Connector Setup #9537

The SuperOffice Administration client will start by using the setup functions to set up the connection to the ERP system. Admin calls **GetConfigurationFields**, and uses the result to add fields to the configuration dialog.



For example: if the call returns a list like this, we get the following dialog:   
[ { “ERP Server URL”, String }, { “Client”, Integer }, { “Token”, String }, { “ERP Parameter”, List } ]



The values are entered by the user.

Fields added after **GetConfiguration-Fields** call.

Figure 2: Please notice that this dialog has been revised. It will look very different in the final version, but the main point is the same.

The user clicks the TEST button in the dialog, and the **TestConnection** method is called with the values from the dialog. When the user clicks OK the values are saved to the SuperOffice database as a connection.

|  |  |
| --- | --- |
| Dictionary< string, FieldMetadataInfo > **GetConfigurationFields**() | Used by the admin client. This is a request for metadata needed to populate the Quote connection configuration admin dialog that takes in the information needed to create a connection to an ERP system. The values entered in the dialog are stored in SuperOffice db and used when **InitializeConnection** is called by the client.  The key in the Dictionary is the FieldKey, and must match the key in the FieldMetadataInfo. |
| PluginResponseInfo **TestConnection**( Dictionary<string, string> connectionData connectionData ) | Used by the admin client. Testing if the connection data is sufficient to get a connection with the ERP system. The Connector should try to do some operations to check if the connection has sufficient rights to run. The connection has not been created yet.  **TestConnection** is called without **InitializeConnection** being called first.  The key in the Dictionary is the FieldKey, and must match the key in the FieldMetadataInfo. The value is what the user entered; **see the paragraph “Config Values” below for details.** |

**Pitfall:**

Note that you cannot populate lists based on the partially filled out user-interface.

Dropdown lists are fetched when the GUI is constructed, so having a configuration GUI like this:

WebService URL:

Dataset:

http://example.com/soap

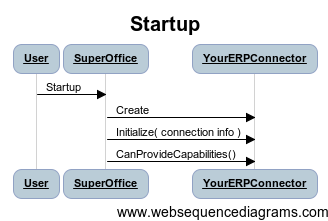
v ………………………………………………

Won’t work – because the call to fetch the Dataset list will come before the WebServiceURL field is filled in.

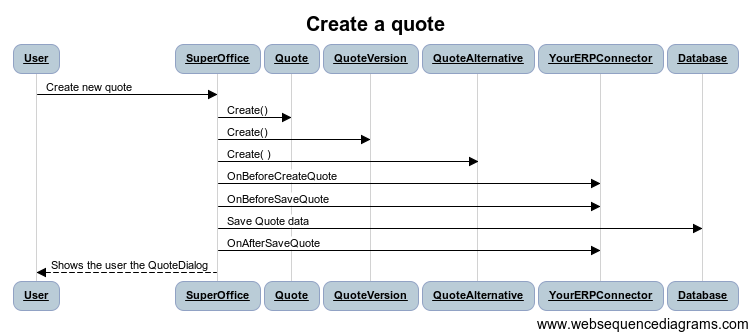
## Basic Connector #9538

The SuperOffice windows client calls **InitializeConnection** during startup using the values saved from the Create connection dialog in the Admin client, and then checks return value to see if the ERP system is available.

If the ERP system is available, SuperOffice will call for instance the OnQuoteCreated when a quote is created.



|  |  |
| --- | --- |
| Int CRMConnectionId | The id of this connection in the CRM system. |
|  |  |
| PluginResponseInfo **InitializeConnection**( QuoteConnectionInfo connectionData, Dictionary<string, string> configurationFields, IProductRegisterCache productRegister) | Set up the connection to the ERP system.  Will be called as part of SuperOffice client startup for each installed connection.  Configuration data comes from the config dialog shown in the Admin client (see **IQuoteConnectorSetup.GetConfigurationFields**)  The key in the Dictionary is the FieldKey, and must match the key in the FieldMetadataInfo. The value is the user entry, as described in **Config Values**.  Return value: **IsOk** set to false if connector can’t provide service (no network); text will explain to user. IsOk = False means the connection is not available, and quotes based on this connection cannot be edited. |
| Dictionary<string, PluginResponseInfo> **GetCapabilities**() | Return a set of capability name status pairs that tell the system what capabilities this connector provides.  Using the PluginResponseInfo gives the connector the possibility to disable a capability, with a reason string that might be shown to the user. |
|  |  |
| QuoteVersionResponseInfo  **OnBeforeCreateQuote**( QuoteContextInfo context ) | Called when a user is creating a quote. The Quote does not exist in database at this time; any changes in the returned QuoteContextInfo will be saved and the GUI updated.   The following parts of the QuoteContextInfo can be updated by this method: Quote; QuoteVersion; QuoteAlternative. Changes to other parts of the QuoteContextInfo will be ignored.  Returns an updated context. |
| void **OnAfterSaveQuote**( QuoteContextInfo context ) | Called after a sale containing a quote is saved: after quote is created, and after quote dialog is closed. (Notice that new items have now gotten their ids in the CRM system.) Changes to the QuoteContextInfo are ignored. |
| void **OnBeforeDeleteQuote**( QuoteInfo quote, ISaleInfo sale ) | Called before a sale containing a quote is deleted. Clean up in the ERP system, if needed. |



### Capability Names

|  |  |
| --- | --- |
| iproductprovider\_provide\_cost | Can the Product Provider fill in the Cost price value?  Determines if the Cost field is shown in the GUI. |
| iproductprovider\_provide\_minimumprice | Can the Product Provider fill in the Minimum price value? Determines if the Minimum Price field is shown in the GUI. |
| iproductprovider\_provide\_stockdata | Can the product provider supply Stock data? Determines whether the Stock values are shown in the GUI or not. |
| iproductprovider\_provide\_extradata | Does the Product Provider fill in any ExtraData. Determines if the extra data fields are shown in the GUI. |
| iproductprovider\_provide\_thumbnails | Can the Product Provider supply thumbnails of products? |
| iproductprovider\_provide\_picture | Can the Product Provider supply any pictures? Determines if the picture field is shown in the dialog. |
|  |  |
| ipriceprovider\_compute\_price | Can the connector calculate a price value for a quote? If the ERP system is not available (e.g. on travel) then the connector might fall back on the list price. |
|  |  |
| iorderconsumer\_place\_order | Can the connector place orders? If not then the PlaceOrder method is not called. The ACCEPT button and the PlaceOrder dialog is still shown, but the connector is not called. |
| iorderconsumer\_provide\_orderstate | Can the connector check order status in the ERP system. If the capability is FALSE, then no GET STATUS button appears after an order has been accepted. |
|  |  |
| ilistprovider\_provide\_productcategorylist | Can the connector provide the Product category list? |
| ilistprovider\_provide\_productfamilylist | Can the connector provide the Product family list? |
| ilistprovider\_provide\_producttypelist | Can the connector provide the Product type list? |
| ilistprovider\_provide\_paymenttermslist | Can the connector provide the Payment terms list? |
| ilistprovider\_provide\_paymenttypelist | Can the connector provide the Payment type list? |
| ilistprovider\_provide\_deliverytermslist | Can the connector provide the Delivery terms list? |
| ilistprovider\_provide\_deliverytypelist | Can the connector provide the Delivery type list? |
|  |  |
| iconnector\_perform\_complexsearch | Can the connector perform a complex search? Will make the UI show the advanced find button. |
|  |  |
| iaddressprovider\_provide\_addresses | Can the connector provide addresses at all? |
|  |  |

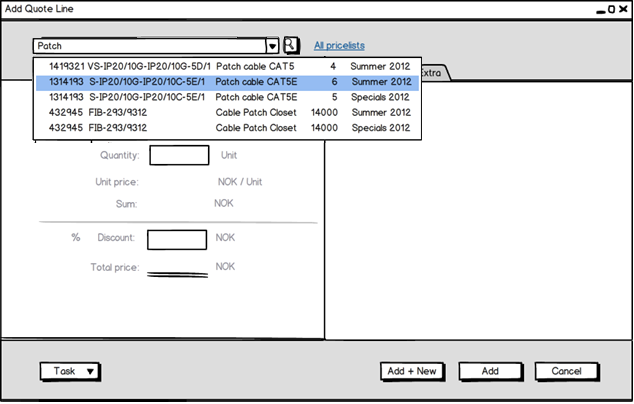
## Product Provider #9539

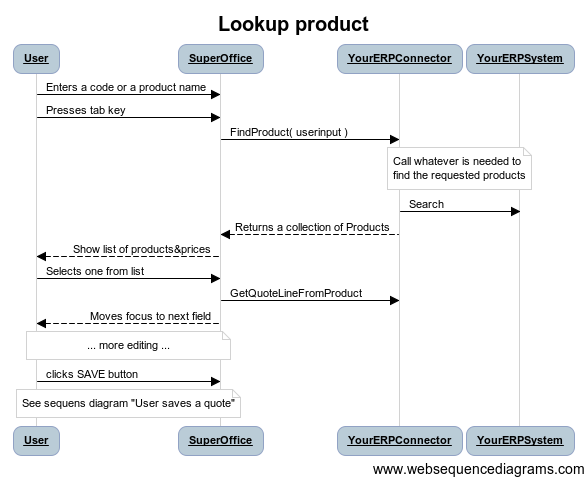
This part searches the ERP system’s product and price list information.

If an ERP system does not provide products, or if the ERP system is not available, the searches can be delegated to the built-in SuperOffice product registry by overriding the QuoteConnectorExtender class.

Currencies are specified in ISO three letter codes: USD, NOK, SEK, EUR, GBP, etc.  
See <http://www.currency-iso.org/dl_iso_table_a1.xls> for details.

The price lists are accessed from the sales & marketing client.

  
*Client uses FindProduct to do simple searches. You can set filters on the available pricelists.*



|  |  |
| --- | --- |
| int **GetNumberOfActivePriceLists**(string isoCurrencyCode ) | Is used to warn the user if there is no active pricelists in a given currency.  Iso currency code like: USD or NOK. Case insensitive.  Will return no of all active pricelists if isoCurrencyCode is empty. |
| PriceListInfo[] **GetActivePriceLists**( string isoCurrencyCode ) | Used by SuperOffice to provide filters. Gets the available active PriceLists in a specific currency.  Iso currency code like: USD or NOK. Case insensitive.  Will return all pricelists if isoCurrencyCode is empty.  Return an empty array if there is no PriceList with the stated currency available. |
| PriceListInfo[] **GetAllPriceLists**( string isoCurrencyCode ) | Currently not used. Gets the all PriceLists in the given currency, including those inactive.  Iso currency like: USD or NOK. Case insensitive. See <http://www.currency-iso.org/dl_iso_table_a1.xls> for details.  Will return all pricelists if isoCurrencyCode is empty. Will return empty array if there is no PriceList available. |
|  |  |
| ProductInfo[] **FindProduct**( QuoteContextInfo context, string isoCurrencyCode, string userinput, string priceListKey ) | The connector should treat this as a free text search; the user might want to search for name, description, product code, extra fields, etc.  Since the return list is a potentially large return value, the connector or the ERP system should limit the number of matches returned to a few hundred.  The dropdown fast searcher calls this function.  Iso currency like: USD or NOK. Case insensitive. See <http://www.currency-iso.org/dl_iso_table_a1.xls> for details.  If the priceListKey is empty, the function will search in all active pricelists.  An empty search (“”) should return null, but a search on “%” should return all products. |
|  |  |
| ProductInfo **GetProduct**( string erpProductKey ) | Gets a product based on erpProductKey.  (If the product is not found, the function will throw an ArgumentException.)  (If the argument is null or empty, the function will throw an ArgumentException.)  Returns the product with the specified key. |
| ProductInfo[] **GetProducts**( string[] erpKeys ) | Return products based on an array of unique ERP keys; handy when you’ve found products through archiveproviders or other mechanisms that leave you holding multiple ERPKey values. |
|  |  |
| QuoteLineInfo **GetQuoteLineFromProduct**( QuoteContextInfo context, string erpProductKey ) | Given a product ERP Key, return a quote line with some default values filled in.  The quoteLineId will be provided by SuperOffice later.  Return the QuoteLine with the product info filled in.  If the product isn’t found, the function will throw an exception. |
|  |  |
| int **GetNumberOfProductImages**( string erpProductKey ) | Currently not used.  Gets the number of images available for this product. |
| string **GetProductImage**( string erpProductKey, int rank ) | Gets the full size picture of the given product.  Rank: Which of the images to return, will in the first version only ask for the first.  Returns the full size picture of the given product. Return NULL if no picture available. |

## Product Search Provider

There are two ways a product can be searched for – the simple, keyword driven search, where the user simply types one or more words; and a more structured search, where a “Find” dialog is shown. This dialog follows the usual convention where users can add one or more criteria, perform repeated searches, and finally choose one or more lines from the results to take back into the Quote system as Quote lines.

The simple search is part of the Product Provider part, specifically the FindProduct method. Implementation of this part and method are mandatory, though the connector is free to interpret the search keywords in whatever manner it sees fit.

The structured search is optional and can be implemented in one of two ways: Either through the optional Product Search Provider functions, or by implementing an IArchiveProvider with a name according to a naming convention. The first option is simpler and more structured, while the second option gives the ultimate flexibility.

A connector signals that it supports this kind of search by supporting the capability iconnector\_perform\_complexsearch. If it contains an appropriately named archive provider, then this will be instantiated; otherwise it must implement the Product Search Provider functions.

#### Using Product Search Provider functions

The Product Search Provider part contains two methods that need to be implemented:

|  |  |
| --- | --- |
| FieldMetadataInfo[] **GetSearchableFields**() | Called before the search dialog is opened, to determine the set of searchable fields (possible criteria).  The connector can offer any fields it desires. If a “current pricelist” is implied by the GUI, then it will be included in the criteria, so connectors must always be prepared to handle erpPricelistKey=xxx. |
| ProductInfo[] **GetSearchResults**( SearchRestrictionInfo[] restrictions) | The actual search: An array of criteria is passed in, and an array of ProductInfo results is returned. The ErpProductKey field of each returned ProductInfo **must** be populated; and as many as reasonable of the other fields **should** be populated. The user can choose which return fields to display. |

The SearchRestrictionInfo structure consists of a criterion name, an operator, and an array of values. The criteria names will be those that the connector specified as available in a preceding GetSearchableFields call (the FieldKey). Operators for all data types are: =, !=, <, >, <=, >=, **between**, **in**. Additionally, strings can use the **begins** and **contains** operators, where % is the 0-or-more wildcard character in the string value.

Values are encoded using the CultureDataFormatter mechanism. The **between** operator uses two values, the **in** operator uses 1..N values, and all other operators just use one value. There are no “magic values”; if the user has chosen “this week” in the GUI for a date field, then the connector will see the **between** operator and the two correct datetime values denoting the start and end of the week.

If there is more than one restriction, then there is an implicit **AND** between them. There is no way to specify **OR** or precedence (parentheses).

#### Using an archive provider

If a connector wants to **return more fields** than exist in the ProductInfo structure, or have custom grouping/icons or richer data types in the criteria, then it can implement an archive provider instead of implementing Product Search Provider functions.

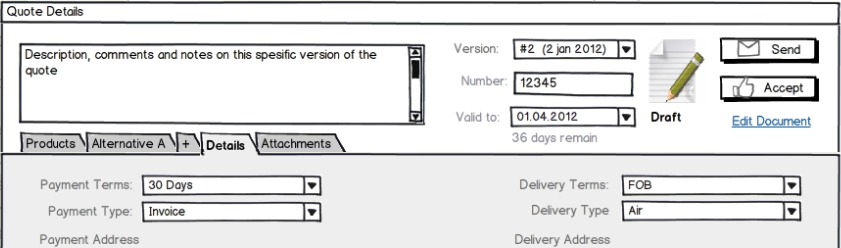
Such a provider must have the name FindErpProduct;<name of connector>, where name in both cases refers to the name in the attribute, not the name of the actual implementing class. Whenever such a provider is detected, then it will be called, following the usual protocol.

This means that it is instantiated twice – once so that GetAvailableColumns and GetAvailableEntities can be called; and once so that SetRestrictions, SetDesiredColumns, SetDesiredEntities, SetPagingInfo and GetRows can be called to perform the actual search.

The provider **must** declare the erpProductKey column and return a valid value. It must also set a LinkHint on the row, of the form “nav=erpProduct&erpProduct\_id=<actual key>” on each row.

Apart from this, it **should** support a reasonable set of columns, which can be a superset of the ProductInfo items.

## Quote List Provider #9540



There are a few lists in the ERP system that we would like to show to the users: payment terms and types, delivery terms and types, and product classifications (product category, product family and product type).

These lists can be supplied by the ERP connector using this function. SuperOffice will take these values and convert the simple flat list of values into a SuperOffice list that appears in the GUI.

If the ERP connector wants to supply a more complex nested list, then the ERP connector can implement a full MDO Provider.

|  |  |
| --- | --- |
| QuoteListItem[] **GetQuoteList**( string quoteListType ) | Gets a named list from the connector  The quoteListType parameter is case insensitive.  Return array of QuoteListItems. Return NULL if the given list is not supported. |

### Quote List Names

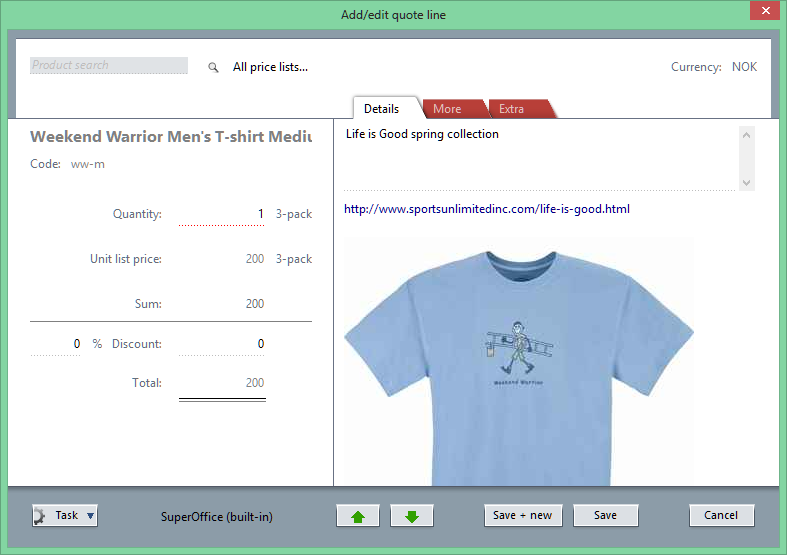
There are some lists in the system we would like the ERP system to provide data for, if it can:

* ProductCategory
* ProductFamily
* ProductType
* PaymentTerms
* PaymentType
* DeliveryTerms
* DeliveryType

If a quote list is NULL, then the GUI will fall back to a text input field, where the user can enter text. This text is passed to the ERP plugin unchanged.

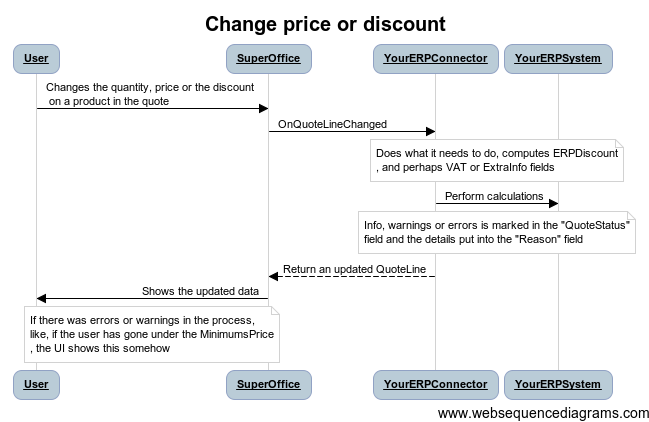
The Quote configuration API may also refer to custom list names which will be filled in by asking here. I.e. you will be asked for more lists than just the ones mentioned here, if you have added custom lists to the configuration dialog.

## Price Provider #9541

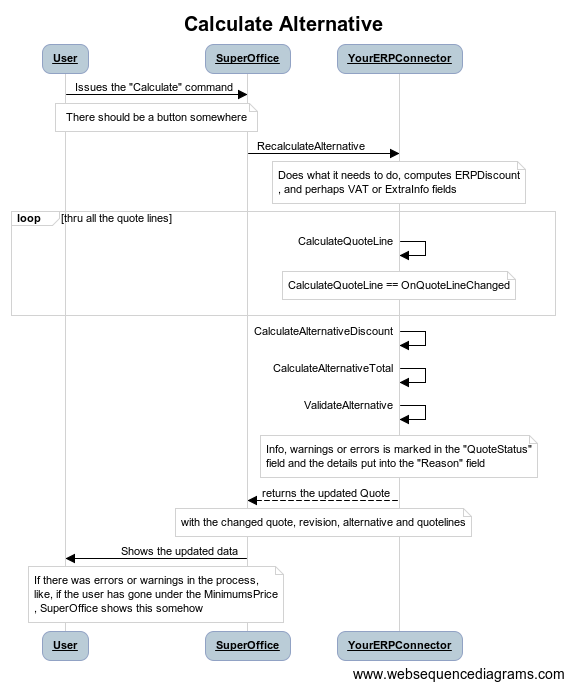


Some installations will be able to compute a price by various means (customer frame agreements chief among these). These calculations are presumably handled by the ERP system. SuperOffice asks the connector to handle changes to the QuoteLine. The ERP connector requests calculations from the ERP system and updates the Quote information in the CRM system.

The ERP Connector is responsible for performing the calculations when the user changes values in the quote, like quantity in a quoteline, or Earning on the quote alternative.

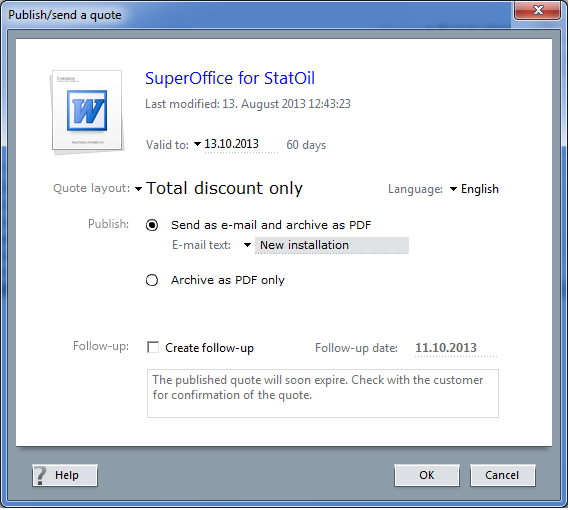


|  |  |
| --- | --- |
| QuoteLineInfo **OnQuoteLineChanged**( QuoteContextInfo context, QuoteLineInfo ql, string[] changedFields ) | Called when the user has changed a field in the Quote Line. The QuoteContextInfo is readonly; QuoteLine may be changed in the return value.  Response time must be fast since this method is called often (every time a field is changed). |
| QuoteAlternativeWithLinesInfo **RecalculateQuoteAlternative**(  QuoteAlternativeContextInfo context ) | The user is finished with entering the quotelines, and wants to calculate the order discount (alternative discount) on this alternative. This method is called whenever the quote lines are changed, or when the user clicks the RECALCULATE button.  QuoteLines and Alternative fields can be changed in the return value.  QuoteVersion, Quote, Sale, Associate and Contact are read-only.  The connector may signal problems with the quote by setting the Quote Alternative Status. |
| QuoteVersionResponseInfo **ValidateQuoteVersion**(  QuoteVersionContextInfo context ) | The user is finished with entering the QuoteLines, and wants to prepare the sending of the quote.  This method is called whenever the user clicks the the Send button or the Place Order.  Quote Lines, Alternatives, Version and Quote fields can be changed in the return value.  Sale, Associate and Contact are still read-only.  A draft quote version will have state = DraftNotCalculated when called. The connector should set the version state to DraftCalculated if the calculations were successful. Leave the state as DraftNotCalculated if the ERP system was not available or some other factor that made the calculation unsuccessful.  The connector can trigger the approval workflow by setting the state to DraftNeedsApproval. When a user with the approval permission has approved or rejected the quote, the quote version state will be DraftApproved or DraftApprovalRejected.  Note that recalculate may also be called when the quote is Approved, or Archived. In these cases, please leave the quote version state alone.  The connector may signal problems with the quote by setting the Quote Version, Quote Alternative or a quote line’s Status to Error, Warning or OkWithInfo, and fill in the Reason field with an explanation. |
| QuoteVersionResponseInfo **UpdateQuoteVersionPrices**(  QuoteVersionContextInfo context ) | The user knows that the prices have changed in the price list, and would like to update the quote with the latest product information from the pricelist.  This method is called whenever the user clicks the the UPDATE PRICES button in the Quote dialog.  Quote Lines, Alternatives, Version and Quote fields can be changed in the return value.  Sale, Associate and Contact are still read-only.  The connector should look up the product in the pricelist and update each pricelist with new prices and other relevant details.  The connector may signal problems with the quote by setting the Quote Version, Quote Alternative or a quote line’s Status to Error, Warning or OkWithInfo, and fill in the Reason field with an explanation. |



A default implementation of price calculations are found in the plugin’s QuoteCalculation class.

## Sending Quotes



When a user sends a quote to the customer, the quote version is frozen, and the quote version cannot be edited further.

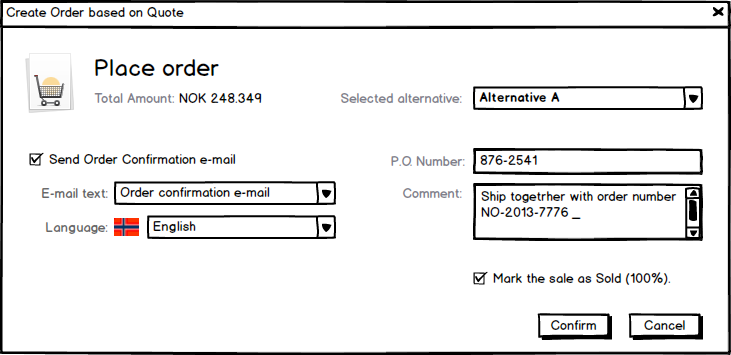
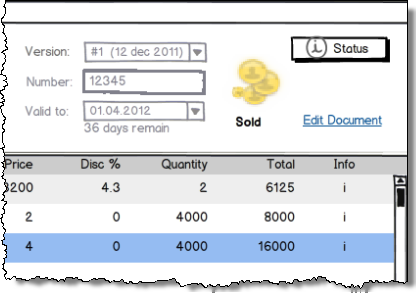
This happens outside the Quote Connector. SuperOffice will generate a PDF document containing the overview document and the order details, and generate an e-mail to the customer if asked.

The connector is not involved in this process.

The QuoteConnector is called after all this is done, so that the connector can sync information about the published quote to the ERP system.

|  |  |
| --- | --- |
| QuoteSentResponseInfo **OnAfterSentQuoteVersion**( QuoteContextInfo context ) | Called after a quote version is sent to the customer. Can return a URL or SO Protocol, or a status message to indicate success/failure. Cannot prevent the version becoming archived. |

## Order Consumer #9542

The user clicks the PLACE ORDER button. The quote version is validated first. If the validation was ok, then the Place Order dialog appears and the user selects the quote alternative to send to the ERP system and clicks OK in the dialog.

Some ERP systems will be able to turn quotes into orders.

If the connector has the iorderconsumer\_place\_order capability, then the connector’s PlaceOrder method is called.

If the connector doesn’t have the iorderconsumer\_place\_order capability, then the quote is just marked SOLD.

After the Quote has been accepted/sold, then the user can check the delivery status with the ERP system.

|  |  |
| --- | --- |
| PlaceOrderResponseInfo **PlaceOrder**( QuoteContextInfo context ) | Place the order in the ERP system.  If the operation returns successfully, the Quote will be locked (completed) in the CRM system, and all updates will come from the ERP system thru the GetOrderState function.  A summary of all the problems (if any) should be placed in the response object. Requires that the iorderconsumer\_place\_order capability is true. |
| OrderResponseInfo **GetOrderState** ( int quoteAlternativeId ) | After the order is created in the ERP system and the user wants to see what the current state of the order is.  Should return a new version with a new alternative and quotelines describing the current state.  This new version will be displayed in the GUI.  If nothing has changed it should return null. Requires that the Order-Status capability is true. |

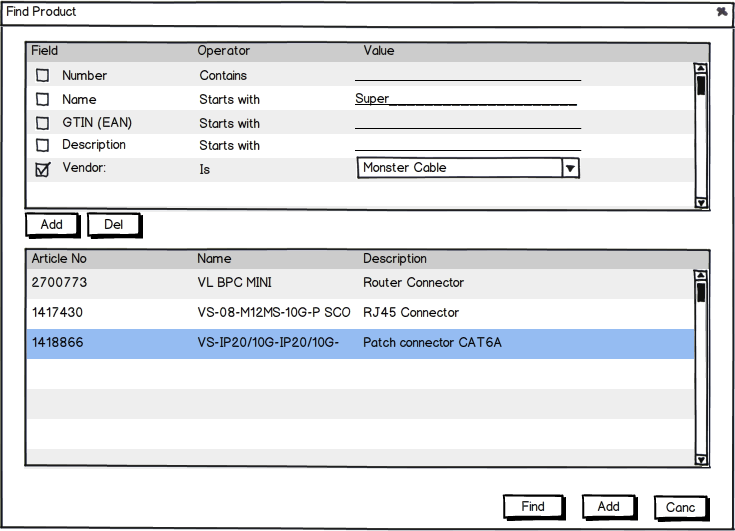
## Address Provider

The ERP systems often has addresses, and when creating a quote, the CRM system calls these to functions to get the default addresses (the user can change the addresses later, if they should happen to be wrong).

|  |  |
| --- | --- |
| AddressInfo[] GetAddresses(  QuoteAlternativeContextInfo context ) | Gets both the invoice and the delivery addresses.  This is normally based on the contact, but can also be based on the project.  [0] = the invoice address  [1] = the delivery address.  Always return an array, but the cell will be null if no address was found on one (or both). |

If the connector returns NULL or does not support addresses, then SuperOffice will use the contact’s street and postal addresses as defaults.

## IArchiveProvider



The archive provider and its associated interfaces are used to implement multi-column lists all over the SuperOffice GUI.

|  |  |
| --- | --- |
| GetAvailableColumns | Get the list of columns handled by this provider |
| GetAvailableEntities | Get the list of entities supported by this provider |
| GetReader( string ) | Start the reader and return an IDataReader (which, as we remember, also inherits IDataRecord for access to individual data fields). This provides an alternative, more generic and more standards-based interface to data. Use **either** GetRows **or** GetReader on any particular archive provider instance. |
| GetResultInformation( ) | Get additional information about the result, such as row count or other optional items. This method should be called some time after GetRows, but before Close. |
| GetRows( string ) | Start the query and return an iterator. The .Current property will be a valid ArchiveRow containing one row, as long as a previous call to .MoveNext returned true. This is the standard semantics for an iterator. |
| SetDesiredColumns( string[] ) | Set which columns should actually be returned, must be a subset of the GetAvailableColumns result |
| SetDesiredEntities( string[] ) |  |
| SetOrderBy( ArchiveOrderByInfo[] ) | Set the sort order. Which columns should rows be ordered by. |
| SetPagingInfo( int, int ) | Set the paging properties of the provider. The default is to fetch page zero, of one thousand (1000) rows. A more reasonable page size is probably around 100. Note that the query processing strategy may change for very large pages (more than 1000) and give significantly longer response times. |
| SetRestriction( ArchiveRestrictionInfo[] ) | Set restrictions on which rows should be returned. |
| Close() | Call this method after the last desired row has been read; this gives the provider the chance to close and free any underlying queries |

## IProductRegisterCache #9543

This part is not in use yet.

SuperOffice has its own simple Product table. This table can be used by the ERP connectors as a cache for ERP data.

This interface will be implemented by SuperOffice. The implementation is passed to all connectors as an initialization parameter.

|  |  |
| --- | --- |
| int **CreatePriceList**( PriceListInfo pricelist ) | Creates a pricelist in the SuperOffice database.  Returns the id to the pricelist. |
| ProductInfo[] **InsertProducts**(int pricelistId, ProductInfo[] products ) | Insert a set of products into the SuperOffice database.  Returns the products updated with their new ids. |
|  |  |
| **UpdateProduct**( ProductInfo newProd ) | Update a product in the cache with new information |
|  |  |
| IProductProvider **GetSuperOfficeProductProvider**( int quoteConnectionId ) | Here you can get the data you have inserted into the SuperOffice tables. |
|  |  |
| void **RemovePriceList**( int pricelistId, bool alsoRemoveRelatedProducts ) | Remove a pricelist and any associated products. |
| void **RemoveProduct**( int productId ) | Will remove a product if it is in a pricelist related to the connection, and the pricelist is an ERP copy. |
| int **InsertImage**(int productId, Image img, int rank ); |  |
| void **RemoveImage**( int imageId ); |  |

# Data carriers

The information the system exchanges between SuperOffice and the connectors is packed in what we like to call “carriers”. These are data-transfer-objects (DTOs). They have no methods/behavior.

The database schema is very similar to these data carriers, but it is not identical. For instance, a pricelist in the database has a PricelistId field; this is not in the carrier, because the carrier shall carry data from other systems as well. So the PricelistId field is put into the ERPPricelistKey field as a string by the SuperOffice connector.

All these carriers are defined in the **SuperOffice.Plugins.DLL** – they will not change after release, unlike the implementation classes stored in **SoDatabase.DLL.**

To keep these classes apart from the internal SuperOffice classes we have suffixed the class names with “Info”, like “QuoteAlternativeInfo”.

## QuoteConnectionInfo #9544

A Quote Connection is set up in the SuperOffice Admin client. It collects the parameters needed to talk to a single ERP client, and gives it a name and an id.

Quote connections will be stored in the table “QuoteConnection” in the CRM database.

|  |  |
| --- | --- |
| int CRMConnectionId | Primary key in the CRM database.  Definition of a connection to an external system, for the Quote system. |
| string ERPName | Name of the ERP system (programmatic). |
| string ERPClientName | Name of the client (company) in the ERP system |
| string ERPClientKey | The identifier for the client in the ERP system. |
| String DisplayName | Connection name shown to user; multi-language support.  The name of the connector to display in a list so that the users can choose between them.  Typically the name of the client, with maybe the ERP system in parenthesis. |
| string DisplayDescription | Tooltip/description shown to user; multi-language support.  Any other info available that would make an uncertain user chose the right connector.  Typically, used for tooltip. |

## ISaleInfo



Figure 5: How Sale and Quote conceptually fit together

Read-only sale information.

|  |  |
| --- | --- |
| int SaleId | Primary key of the sale record in CRM database |
| string Associate | Owner of the sale |
| string Number | Sale number |
| string Title | Title of the sale |
| double Amount | Total sale amount |
| string Currency | Sale currency name |
| string PublishStartDate | Sale, start date for publishing |
| string PublishEndDate | Sale, end date for publishing |
| string Type | Sale type - an MDO list item name |
| string Stage | Sale stage - an MDO list item name |
| string Credited | Credited to - an MDO list item name |
| string Competitor | Main Competitor for sale - an MDO list item name |
| short Probability | Probability percent - default derived from Stage list |
| string NextDueDate | Date of the first uncompleted activity. |
| string Reason | Reason the sale is lost/sold etc. |
| string SaleDate | Expected closing date |
| string Status | Open/Sold/Lost/Stalled |
|  |  |
| Decimal Cost | Total cost |
| Decimal Profit | Total profit (Amount – Cost) |

## QuoteInfo

A Sale can have a quote, and have then a 1-1 relationship with the QuoteInfo.

Read-write Quote information.

|  |  |
| --- | --- |
| int QuoteId | Primary key in CRM database. |
| int SaleId | The foreign key to the corresponding sale. |
| int QuoteConnectionId | The connection in the CRM system to where this quote came from.  Identifies the ERP connection used for this quote. Each quote is bound to one and only one connection. |
|  |  |
| string ERPQuoteKey | Foreign key of quote (if available).  The key in the ERP system that identifies this sale's Quote (as opposed to the later Order information) |
| string ERPOrderKey | The key in the ERP system that identifies this sale's Order, as transferred and possibly later edited in the ERP system.  Only filled out if there exists a corresponding order representation of the quote in the ERP system. |
|  |  |
| int ActiveQuoteVersionId | The primary key of the Quote Version that is currently active. (The active version will always be the latest version.) |
| int AcceptedQuoteAlternativeId | The primary key of the Quote Alternative which was finally accepted by the customer.  Set when the user is marking a quote as accepted. |
| int DocumentId | The ID of the main Quote Document. This is not the document containing the products, but the other one. |

## QuoteVersionInfo

Represent a version of a quote

A quote is divided into one or more versions (or revisions, if you like), so a quote have 1..n QuoteVersions.

A quote is divided into one or more versions (or revisions, if you like), so a quote have 1..n QuoteVersions.

I.e. a QuoteVersion always have a quote.

Even if versioning is disabled, a single version will exist. When versioning is disabled, new versions are not created, but the only one is reused.

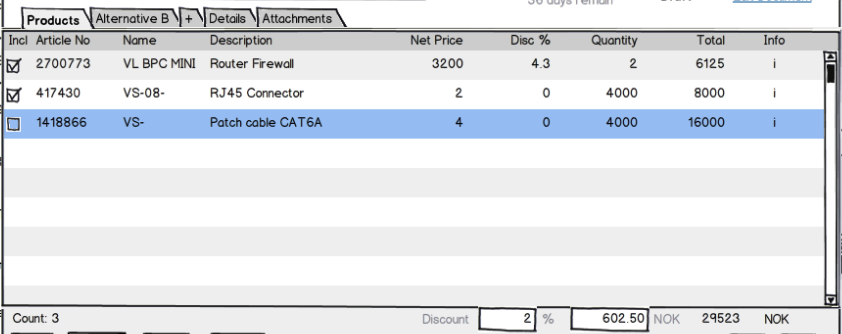
|  |  |
| --- | --- |
| int QuoteVersionId | Primary key in CRM database. |
| string ERPQuoteVersionKey | Key in the ERP system that uniquely identifies this Version within the ERP system (if available, the field may be empty). |
|  |  |
| int QuoteId | Foreign key to CRM quote (the conceptual parent).  Owning Quote of this Quote Version |
|  |  |
| string Description | Description of Version.  Potentially longer text description, typically used in a tooltip.  Max 2K. |
| string Number | A quote number that the user (or ERP connector) can fill out. |
| QuoteVersionStateInfo State | Current state of this quote version.  The states will be like: CalculatedDraft, NotCalculatedDraft, Published, etc. |
| int LikelyQuoteAlternativeId | The alternative that is considered most likely to be accepted.  Used to calculate probable income. |
|  |  |
| DateTime SentDate | The date the version was sent to the customer. |
| int FollowupId | Link to a follow-up activity, created when this quote version was sent to the customer. |
| DateTime ExpirationDate | Last date the quote Version is valid, expiration is at midnight end of this day. |
|  |  |
| string DeliveryCountry | The quote has an address for delivery. Should be stored as ISO code or something… |
| bool HasOwnDeliveryAddress | The delivery address is not the same as the contact's Street address |
| string InvoiceCountry | The quote has an address for Invoicing. Should be stored as ISO code or something… |
| bool HasOwnInvoiceAddress | The quote has an address for Invoicing. This will typically be copied from the company's addresses. |
|  |  |
| string ERPPaymentTermsKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text.  For instance: ‘Standard 30 days’. |
| string ERPPaymentTypeKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text.  For instance: ‘Invoice’. |
| string ERPDeliveryTermsKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text.  For instance: ‘FOB’ (‘Free on board’). |
| string ERPDeliveryTypeKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text.  For instance: ‘Air’. |
|  |  |
| int Rank | Rank/Version number, starts at 1. |
|  |  |
| QuoteStatus Status | If there was a problem with for instance calculation, this field is set to warning or error. |
| string Reason | If there was a problem, this field contains a localized explanation of the problem and possible steps to fix it that the user can be shown. |
|  |  | |
| string ExtraField1 | Optional information added by Quote Connector; usable in the quote document merge process. | |
| string ExtraField2 | Optional information added by Quote Connector; usable in the quote document merge process. | |
| string ExtraField3 | Optional information added by Quote Connector; usable in the quote document merge process. | |
| string ExtraField4 | Optional information added by Quote Connector; usable in the quote document merge process. | |
| string ExtraField5 | Optional information added by Quote Connector; usable in the quote document merge process. | |
|  |  |
| int ApprovedBy | Not yet implemented:  Id of associate who approved (or rejected approval) for this version. |
| string ApprovedText | Not yet implemented:  Text with comments on why approval was granted (or rejected) |
| int ApprovedRegisteredBy | Not yet implemented:  Id of associate who actually entered the approval; might be different from ApprovedBy (i.e. due to telephone consultation/approval) |
| DateTime ApprovedRegisteredDate | When was approval granted or rejected |
| DateTime LastRecalculated | When this version was last subjected to a total recalculation. This field must be set by the connector, since the connector may choose to ignore a RecalculateVersion call based on policies and possibly the current value of this field. SuperOffice will set this field to 1.1.1760 whenever any change occurs to the quote, to indicate that a recalculation is needed. |

### QuoteVersionStateInfo

The various states a QuoteVersion can be in.

|  |  |
| --- | --- |
| Unknown | State unknown |
| DraftNotCalculated | This is a draft that has not been calculated |
| DraftCalculated | Draft that has been calculated, and ERP has verified it as OK |
| DraftNeedsApproval | Draft has been checked, and there was a problem that needs approval |
| DraftApproved | Draft with potential problems has been human-approved |
| Sent | Sent to customer, and is presumably a legally binding document |
| SentExpired | Sent to customer, but has expired and is no longer binding |
| Archived | Archived without being sent |
| Ordered | Accepted and ordered by customer |
| Rejected | Version was rejected be the customer |
| Sold | Quote was accepted and the sale was carried through all steps |

## QuoteAlternativeInfo



Quote Version is made up of one or more Alternatives. One of 1..n possible alternatives in a Quote Version.

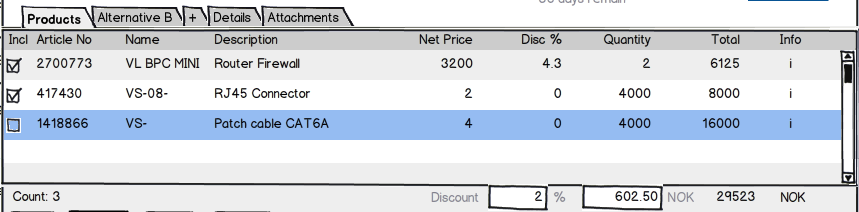
The reason we have alternatives is that a quote can say to a customer, “we can solve you problem in two (or more) different ways, with different technology and side effects (and price)”.

An Alternative may have discounts on the total amount. The Alternative tracks whether the user on the order level entered the Discount %, the Discount amount, the Earning%, Earning amount or the TotalPrice fields so that the discount and earning and total can be re-calculated correctly when Quote Lines are added or changed.

|  |  |
| --- | --- |
| int QuoteAlternativeId | Primary key to the Alternative in the CRM system. |
| string ERPQuoteAlternativeKey | Key that identifies this alternative in the ERP system, if it exists there. |
|  |  |
| int QuoteVersionId | The version that owns this alternative (the chain is Sale 1->1 Quote 1->+ QuoteVersion 1->+ QuoteAlternative. |
|  |  |
| string Name | Name of Alternative. Shown in tab in user interface. |
| string Description | The tool-tip to use in the user interface (on the tab, for instance). |
|  |  |
| QuoteStatus Status | If there was a problem with for instance calculation, this field is set to warning or error. |
| string Reason | If there was a problem, this field contains a localized explanation of the problem and possible steps to fix it that the user can be shown. |
|  |  |
| double ERPDiscountPercent | The discount the system calculates based on customer /amount / whatever. Can be overridden by the salesman in the field ‘DiscountPercent’ or ‘DiscountAmount’.  Both the two ‘ERPDiscountPercent’ and ‘ERPDiscountAmount’ shall be filled out.  If UserValueOverride is 'None', then the ERPDiscountAmount shall be copied into DiscountAmount and ERPDiscountPercent into DiscountPercent.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| double ERPDiscountAmount | The discount the system calculates based on customer /amount / whatever. Can be overridden by the user in the field ‘DiscountPercent’ or ‘DiscountAmount’.  Both the two ‘ERPDiscountPercent’ and ‘ERPDiscountAmount’ shall be filled out.  If UserValueOverride is 'None', then the ERPDiscountAmount shall be copied into DiscountAmount and ERPDiscountPercent into DiscountPercent. |
|  |  |
| double DiscountPercent | The discount the salesman specifies, in percent.  Both the two ‘DiscountPercent’ and ‘DiscountAmount’ shall be filled out, but the UserValueOverride field must be set to the field the user actually changed.  If this field is filled out by the user, it overrides the discount suggested by the connector.  If the user has not filled any values, the system will copy the ERP discount % value into this field.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| double DiscountAmount | The discount the salesman specifies, in whatever currency the sale is in.  Both the two ‘DiscountPercent’ and ‘DiscountAmount’ shall be filled out, but the UserValueOverride field must be set to the field the user actually changed.  If this field is filled out by the user, it overrides the discount suggested by the connector.  If the user has not filled any values, the system will copy the ERP discount amount value into this field. |
| ValueOverrideInfo UserValueOverride | Has the pre-calculated (from ERP) price information been overridden, and how.  If the user has filled out the discountpercentage field, then the UserValueOverride field is set to OverridePercent.  (The DiscountAmount, EarningPercent, EarningAmount and TotalPrice fields are calculated based on the discountPercent.) |
|  |  |
| string VATInfo | Extra info about VAT that the connector might insert.  This is just to help out the layout of the quote in a document.  In this field we store VAT info that needs to be printed out on the quote, like “Inc VAT” or “12% VAT”.  May or may not be filled out. |
| double VAT | Tax/VAT if available from ERP system; not used in any business logic.  This is just to help out the layout of the quote in a document, but SuperOffice will not try to calculate this value.  May or may not be filled out. |
|  |  |
| double EarningAmount | Earning on this alternative, as an absolute amount (in money). |
| double EarningPercent | The earning on this alternative, in percent of total.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| double SubTotal | The sum of the quotelines totalPrice (and not the sum of their subtotal!!).  Think of it as sum before discount. |
|  |  |
| double TotalPrice | Sum of the QuoteLines.TotalPrice - AlternativeDiscount  or QuoteLines.TotalCost + Earning  based on what, if anything, the user has entered last.  Shall be calculated by the connector. |
| string ExtraField1 | Optional information added by Quote Connector; usable in the quote document merge process. |
| string ExtraField2 | Optional information added by Quote Connector; usable in the quote document merge process. |
| string ExtraField3 | Optional information added by Quote Connector; usable in the quote document merge process. |
| string ExtraField4 | Optional information added by Quote Connector; usable in the quote document merge process. |
| string ExtraField5 | Optional information added by Quote Connector; usable in the quote document merge process. |

### Quote Alternative Discounts/Earning

Each quote line can have a discount applied. A quote alternative (a set of quote lines) can have a separate discount applied - depending on a company policy preference.



The discount on the whole quote alternative works much like the discounts on the quote.  
The ERP system can suggest a discount (either Percent or a fixed amount) and the user can override the suggestion.

### Cost + Earning = SubTotal – Discount = TotalPrice

Another way to set the discount is to set the earning! Since Cost + Earning = TotalPrice and Discount = SubTotal – TotalPrice.

This means that if you set any of the fields:

* DiscountAmount
* DiscountPercent
* EarningAmount
* EarningPercent
* TotalPrice

Then the other 4 values will be adjusted accordingly.

The QuoteCalculation helper class in the plug-in DLL can help you handle the different methods of calculating totals and discounts.

## QuoteLineInfo #9546

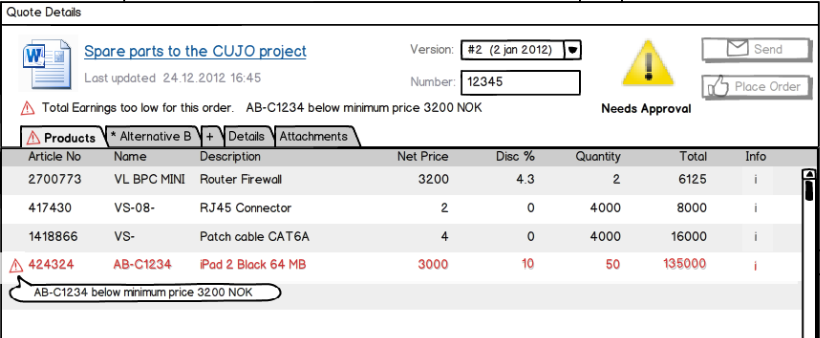


Figure 6: Quote Line archive – list of quote lines in an alternative.

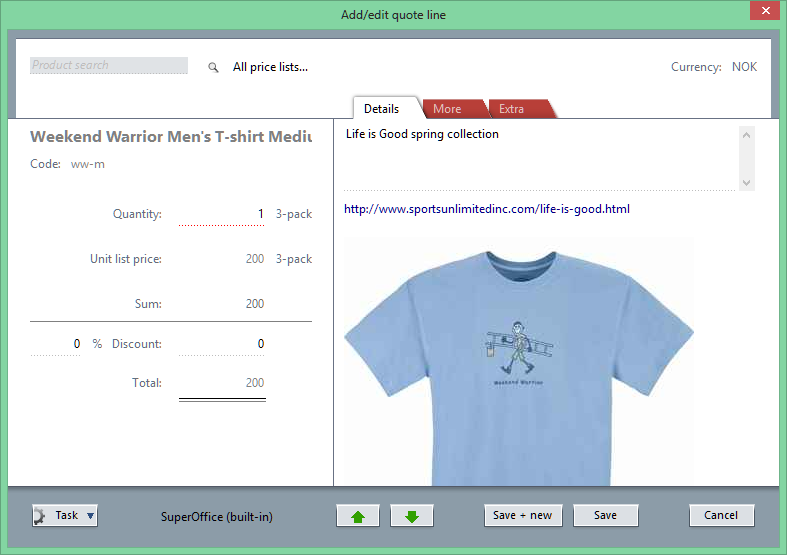


Figure 7: Quote Line dialog – details of one quote line in an alternative.

QuoteLines are mainly information copied from the Products provider. Products information is sometimes edited by the user before being included in the quote, so most information is duplicated from Product rather than referenced directly.

|  |  |
| --- | --- |
| int QuoteLineId | Primary key in CRM database. |
| string ERPQuoteLineKey | The foreign key to the quoteline in ERP system (if it has such a representation). |
|  |  |
| int QuoteAlternativeId | The alternative this line is part of, the conceptual Parent in CRM database. |
|  |  |
| tring ERPProductKey | Foreign key of product this line is based on.  Can be blank since the QuoteLine doesn’t have to be connected to a product. |
| string ERPPriceListKey | Foreign key to the price list that this quoteline is a part of.  Can be blank since the QuoteLine doesn’t have to be connected to a product. |
|  |  |
| QuoteStatus Status | If there was a problem with for instance calculation, this field is set to warning or error.  Typically shown as an icon. QuoteStatus is an enum with statuses: OK, OKWithInfo, Warning, Error. |
| string Reason | If QuoteStatus is not OK, then this field contains a localized explanation that the user can be shown. |
|  |  |
| double Quantity | How many units; this is a decimal field since you might want to offer fractional units (2.5kg, or 0.5PC). |
| double DeliveredQuantity | How many units have been delivered - updated by the ERP system. |
| int Rank | QuoteLines can be re-ordered, so we must track the ordering. |
|  |  |
| string Name | The name of the product.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string Description | A longer description for the product.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string Code | A value the salesmen use to quickly find the correct product.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string QuantityUnit | What is the unit (meter, ton, bushel, microsecond, gradus, τρυβλίον, 五合枡, دونم or whatever); Connector handles conversion relative to PriceUnit if they are different. |
| string PriceUnit | What is the unit (meter, ton, bushel, microsecond, gradus, τρυβλίον, 五合枡, دونم or whatever); read-only for lines that originate in defined products. |
| string ItemNumber | No: «Postnummer». Specific numbers from some hierarchy, for instance “1.4.3.2”.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string Url | A url to the product info. Can be empty.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
|  |  |
| string ProductCategoryKey | Either a List id to an id from a connector provided list, or  , if the connection doesn't support lists, a text.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string ProductFamilyKey | Either a List id to an id from a connector provided list, or  , if the connection doesn't support lists, a text.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string ProductTypeKey | Either a List id to an id from a connector provided list, or  , if the connection doesn't support lists, a text.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
|  |  |
| string SupplierCode | The suppliers' code or part number for this product.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
| string Supplier | The name of the supplier.  Is stored here if the user changes the value from the product in the pricelist, or just enters a QuoteLine without a product link. |
|  |  |
| string Thumbnail | The thumbnail of the product, if it exists.  Base64 encoded string, or a valid URI that resolves to an image. |
|  |  |
| string VatInfo | Tax/VAT information.  Extra info about VAT that the connector might insert, and the users might want to specify on the quote.  This is just to help out the layout of the quote in a document. |
| double VAT | Tax/VAT if available from ERP system; this field is not used in any business logic in SuperOffice.  This is just to help out the layout of the quote in a document, but SuperOffice will not try to calculate this value. |
|  |  |
| double UnitCost | The cost price per unit for this product. May be filled in by connector if it has the Provide-Cost capability. |
| double UnitMinimumPrice | The minimum price this line can be sold for (to limit discounting). Will come from the connector. List price per unit must exceed the minimum price per unit. |
| double UnitListPrice | The standard list price; as given by ERP Connector, OR overridden by user |
| ProductExtraDataFieldInfo[] ExtraInfo | Extra data (fields with labels). Shall be shown in the quoteline dialog.  Additional info that the ERP system would like to store and show in the user interface.  Information placed here is shown in the GUI if the “provide-extra-data” capability is true.  Different products can have different fields.  It will not be possible to directly put info here into the quote document.  BTW, this will be stored in the SuperOffice database as an xml field, like this:  <Fields>  <Field Name="Weight" Type="String"><![CDATA[[F:16.6] tons]]></Field>  <Field Name="Height" Type="String"><![CDATA[ [F:44.0]cm]]></Field>  <Field Name="Arms" Type="String"><![CDATA[ [I:2]]]></Field>  <Field Name="Certification" Type="String"><![CDATA[AB-ICE]]></Field>  <Field Name="Weight" Type="String"><![CDATA40°C]]></Field>  <Field Name="Security info" Type="Url" ><![CDATA[http://www.armystudyguide.com/content/army\_board\_study\_guide\_topics/hand\_grenades/throwing-of-hand-grenades.shtml]]></Field>  <Field Name="Security image ><![CDATA[http://upload.wikimedia.org/wikipedia/commons/thumb/8/80/MK2\_grenade\_DoD.jpg/220px-MK2\_grenade\_DoD.jpg</Field]]>  </Fields> |
|  |  |
| string Rights | Field1=right&Field2=right, etc. of any fields that have non-standard field access rights.  Rights can be one of: N (=None or Hidden), R (=Read-only), W (=Writeable), M (=Mandatory).  The fields will mostly be from the Quoteline table, but some added fields that are conceptually part of the quoteline, like Image will also be possibly to set rights on.  See Rights field for more information  Will be used by SuperOffice to control the user interface when showing the record. |
| string Rule | The names of one or more calculation rules that are in effect for this line, comma-separated case-insensitive.  Will NOT be used by SuperOffice. |
|  |  |
| string ExtraField1 | This a simple field for adding information that the Connector can provide, and that the qoute document need to display. |
| string ExtraField2 | This a simple field for adding information that the Connector can provide, and that the qoute document need to display. |
| string ExtraField3 | This a simple field for adding information that the Connector can provide, and that the qoute document need to display. |
| string ExtraField4 | This a simple field for adding information that the Connector can provide, and that the qoute document need to display. |
| string ExtraField5 | This a simple field for adding information that the Connector can provide, and that the qoute document need to display. |
|  |  |
| double ERPDiscountAmount | The discount the system calculates based on customer / quantity / whatever.  Can be overridden by the salesman in the field 'DiscountPercent' or 'DiscountAmount'.  If UserValueOverride is set to ‘None’ then the value is copied to DiscountAmount.  Both fields ERPDiscountPercent and ERPDiscountAmount will be filled out. |
| double ERPDiscountPercent | The discount the system calculates based on customer / quantity / whatever.  Can be overridden by the salesman in the field 'DiscountPercent' or 'DiscountAmount'.  Both fields ERPDiscountPercent and ERPDiscountAmount will be filled out.  If UserValueOverride is set to ‘None’ then the value is copied to DiscountPercent.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| double DiscountAmount | The discount for the line, in whatever currency the sale is in.  Both ‘DiscountPercent’ and ‘DiscountAmount’ shall be filled out, but the UserValueOverride field must be set to the field the user actually changed last.  If this field is filled out by the user, it overrides any discount suggested by the connector.  If the user has not filled this in, the system will copy the ERP discount amount to this field. |
| double DiscountPercent | The discount for the line, in percent.  Both ‘DiscountPercent’ and ‘DiscountAmount’ shall be filled out, but the UserValueOverride field must be set to the field the user actually changed last.  If this field is filled out by the user, it overrides any discount suggested by the connector.  If the user has not filled this in, the system will copy the ERP discount amount to this field.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| ValueOverrideInfo UserValueOverride | Has the pre-calculated (from ERP) price information been overridden, and how.  If the user has filled out the discountpercentage field, then the UserValueOverride field is set to OverridePercent.  (The DiscountAmount, EarningPercent, EarningAmount and TotalPrice fields are calculated based on the DiscountPercent.) |
|  |  |
| double EarningAmount | The earning, in whatever currency the sale is in.  Both ‘EarningAmount and ‘EarningPercent shall be filled out, but the UserValueOverride field must be set to the field the user actually changed last. |
| double EarningPercent | The earning, in percent.  Both ‘EarningAmount and ‘EarningPercent shall be filled out, but the UserValueOverride field must be set to the field the user actually changed last.  The Percentage is given in percent form, i.e. ‘12%’ is represented as ‘12’. |
| double SubTotal | (UnitListPrice \* Quantity)  Calculated by the ERPconnector |
|  |  |
| double TotalPrice | TotalPrice = SubTotal - DiscountAmount  or  TotalPrice = (UnitCost \* Quantity) + EarningAmount  , according to what the user changed last. |
| bool IsIncluded | Not yet implemented:  “IsNotAnOption”  If true, will be added to the total price.  Shown as a checkbox on the quote line.  Not in V1 |
| bool IsGroupHeading | Not yet implemented:  Indicates that the line is a Group Heading.  The name will be used as Label.  Totalprice will reflect the sum of all totalprices in the quotelines connected to the group.  The rank shall be ascending thru the whole QuoteAlternative, disregarding any groups  Not in V1 |
| int ParentQuoteLine | Not yet implemented:  If this quoteLine is a part of a group heading or a Package, this field will have that quoteline’s id.  Not in V1 |

### Enum ValueOverrideInfo

Which field the user changed last. This helps the calculation to calculate the other fields correctly.

None = 0,

OverrideTotal = 1,

OverrideDiscountPercent = 2,

OverrideDiscountAmount = 3,

OverrideEarningPercent = 4,

OverrideEarningAmount = 5

### Rights field

Specification: “Field1=Right,Reason&Field2=Right,Reason”

***Right:***

* N = None or Hidden
* R = Read-only (Implies Visual)
* W = Writable (& Visual)
* M = Mandatory (& Writable & Visual)

The rights are mutually exclusive; a field can only have one of these rights.

***Field:***

The fields will mostly be from the Quoteline table, but some added fields that are conceptually part of the quoteline, like Image will also be possibly to set rights on.

***Reason:***

The reason is a description that will be used on the field (and label) as a tooltip to explain to the user why this field on this product is different. You don’t have to insert anything here, but a SuperOffice user will expect this.

You cannot use the sign ‘&’ in the description.

The reason will only work when the right is R (Read-only).

***Example:***

“QuoteLine.Image=N&QuoteLine.UnitCost=R,This product has a fixed cost.&QuoteLine.Description=W&QuoteLine.VAT=M”

### QuoteListItemInfo

One line in a list, consisting of a key, a name, a tooltip and an icon. Lists are “flat” with no headings.

|  |  |
| --- | --- |
| String ERPQuoteListItemKey | Primary key for the item |
|  |  |
| String DisplayValue | List item text to display. |
| String DisplayDescription | Typically used in a tooltip. |
| String Icon | Icons can be the names of existing icon files in the system, URL’s pointing to PNG/JPG images, or base64-encoded PNG/JPG images. |

### QuoteVersionResponseInfo

Is returned when a version with all alternatives and lines can be changed by the connector.

|  |  |
| --- | --- |
| QuoteInfo CRMQuote | Quote information |
| QuoteVersionInfo CRMQuoteVersion | Version information. |
| QuoteAlternativeWithLinesInfo[] CRMAlternativesWithLines | Alternative information. |

### QuoteSentResponseInfo

Return value on Quote Sent.

|  |  |
| --- | --- |
| QuoteVersionResponseInfo QuoteData | Quote information |
| string Url | Url to navigate to, if non-blank |

### OrderResponseInfo

Specialized version of PluginResponseInfo for GetOrderState.

Inherits PluginResponseInfo.

|  |  |
| --- | --- |
| QuoteInfo CRMQuote | Quote information |
| QuoteVersionInfo CRMQuoteVersion | Version information. |
| QuoteAlternativeWithLinesInfo CRMAlternativeWithLines | Alternative information. |

### PlaceOrderResponseInfo

Specialized version of PluginResponseInfo for PlaceOrder.

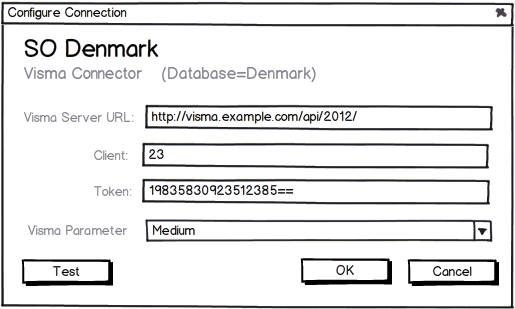
Inherits OrderResponseInfo.

|  |  |
| --- | --- |
| Url | An optional URL; if non-blank then the GUI will navigate to this soprotocol: or http(s): as the final operation of the PlaceOrder. |

### FieldMetadataInfo

This carrier describes a custom field to be added to the config dialog at runtime. It is used to populate the Admin configuration dialog for a connection. Note that this is just a description of the field - it is not the field itself.

The GUI will use this info to build the user interface controls. The call to the ERP Connector’s GetConfigurationFields method returns a list of fields and field types. The Admin client builds a dialog with these fields.



The values are entered by the user.

Fields added after **GetConfiguration-Fields** call.

Figure 3: Please notice that this dialog has been revised. It will look different in the final version.

|  |  |
| --- | --- |
| string FieldKey | Internal name of the field. Used as the key in the dictionary of values. |
| int Rank | A way to set the order of the fields. Lowest value will be displayed first/over the fields with other values. |
| string DisplayName | Localized field name – shown in the GUI as the label for the control. |
| string DisplayDescription | Tooltip for the field. |
| FieldMetadataTypeInfo FieldType | String, int, decimal, etc.  Kind of field widget to use: textbox, number field, password field, dropdown list or checkbox? |
| string ListName | Used for getting the list items from the list provider. The source of the items must ultimately be exposed as an IQuoteListProvider, through various naming conventions and adapters. The actual Quote and ERP Connectors solve this in slightly different ways. |
| string DefaultValue | A default value for the field. The value in the widget when the configure dialog is opened in Add Connection mode.  This will vary a bit depending on the type, of course.  Obviously, if the field is a label, text or password, then the text is used.  If the field is an int or double, we shall try to convert the string into a number. (If the conversion fails, we shall use zero or 0.00 as the default value.)  If the field is a list, we shall try to find a list item where the fields default value matches the listsitem’s ERPQuoteListItemKey. |
| int MaxLength | Maximum length for strings, if set.  0 means no restriction. (Though sooner or later **something** will no doubt overflow if you pile on the gigabytes.) |
| FieldAccessInfo Access | Access restrictions on the field |

### Enum ConfigFieldType

Describes the different types of controls that can appear in the Configure connection dialog:

* Checkbox – checkbox control. Returns 0 or 1
* Text – edit field
* Password - edit field with \*\*\* masking
* Integer – edit field – digits only, accepts integers
* Double - edit field – digits only, accepts decimal numbers formatted with CultureInfo.InvariantCulture (for instance: “-1000.01”)
* List – dropdown list
* Label – static text (no value entered or saved)

### Enum FieldAccessInfo

Access restrictions and mandatory status, if any.

* Normal - Normal field, no particular restrictions
* Mandatory - This field is mandatory
* ReadOnly - This field is read-only

### Config values

Configuration fields can be declared to be one of a number of different types, using the FieldMetadataInfo. However, they are always transmitted as strings; and to do the conversion between strongly typed value and string we use the SuperOffice.Globalization.CultureDataFormatter class. Use the ParseXXX methods from the same class to get back to the correct type (int, datetime, etc).

### PluginResponseInfo

PluginResponseInfo exists to be able to respond with more than just a true/false, but also an explanation. Such an explanation can be displayed on for instance a disabled “Place Order” button.

|  |  |
| --- | --- |
| Bool IsOk | Answer to the question / An indication if the operation went well. |
| String UserExplanation | A localized explanation to the answer.  Example: US:"Text in English\";NO:\"Text in Norwegian\";GE:\"Text in German\"FR:\"Text in French\"; and so on |
| String TechExplanation | Always in English |
| String ErrorCode | An error code, if available. |

### Enum QuoteStatus

QuoteStatus shall be used to give indications in the user interface that there is more info or problems available.

* Ok, // OK, all is good
* OkWithInfo,// All is good, but there is some additional information that the user should be made aware of.
* Warning, // There is a problem that the user must be made aware of.
* Error, // There is a problem that the system will not be able to get around. The user needs to do something. **SuperOffice will deny the user to send the quote or place the order.** If the error is not so bad thet it have to stop the user? Then it is not an error, it is a warning.

Example: The user has registered a quoteline that is discontinued and the amount the user has registered is not in stock.

### Error system

When something is wrong and the connector needs to make the user aware of the problem, there are a few ways to do this, based on what fails.

If the connector throws an exception, SuperOffice will catch it and present the Message to the user. This should be avoided, it could leave the system in a bad state.

The QuoteVersion, QuoteAlternative and QuoteLine objects all have the same structure: a Status and a Reason field. The Status field is of type QuoteStatus, and can thus be “OK, OKWithInfo, Warning or Error”. (See Enum QuoteStatus for details). When setting the fields you should set the state for the level you are in (version, alternative or line), and SuperOffice will concatenate and present the problems to the user in the user interface.[[1]](#footnote-1)

If, when calculating a quoteline, you find a problem with the quoteline (typically the product is discontinued and the stock is empty), you should set the Status field and put a user friendly explanation in the Reason field.

If, when calculating the version, you have a problem connecting to some back-office system (perhaps the user is offline) that you must have access to to be able to calculate, then you should set the QuoteVersion.Status to error and explain the problem in the QuoteVersion.Reason field.

And so on.

##### Reason fields.

The reason fields will be shown to the user, and should thus be translated to the language of the user.

If you deliver the text in the following format, SuperOffice will select the right language for you:

"US:\" A problem occured\";NO:\" Et problem oppstod\";SW:\”Ett problem uppstod\”;GE:\" Ein Problem ist aufgetreten\";FR:\" Un problème est survenu\""

And my apologies for the translations, I used translate.google.com.

A text is multi-language if it follows the layout:

LL:"text";...

LL is a language code of two letters, same as the one used for loading the language resource DLL

The colon is literal.

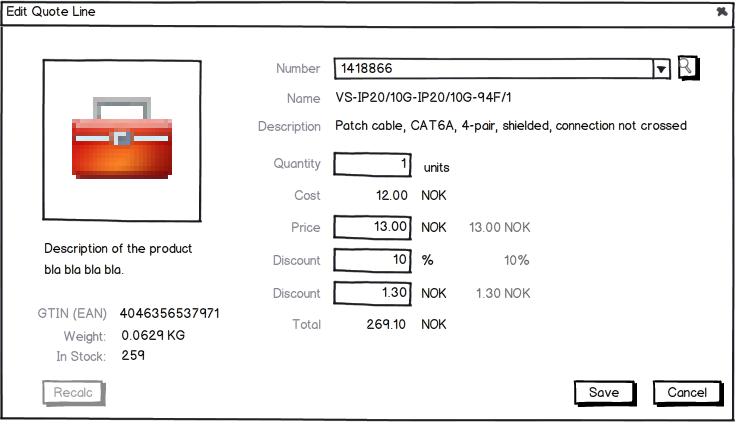
The text is the text for this language, inside double quotes (no quotes inside the text please).

The semicolon is literal.

Please note that the format is quite strict; if the text does not follow these conventions, it is not parsed at all, so do test the errors too.

Some of the functions also returns a PluginResponseInfo. See PluginResponseInfo for more info.

### ERP Discounts and User Discounts



ERP suggested discounts (in percent and total amount)

Figure 8: Please notice that this dialog has been revised severely. It will look very different in the final version.

Each quote line has two discounts: one suggested by the ERP system, and one entered by the user.

If the user does not enter anything, we default to the ERP system suggestions.

If the user enters a discount, the ERP connector gets a chance to change them.

### Value Fields and Amounts and Percent’s

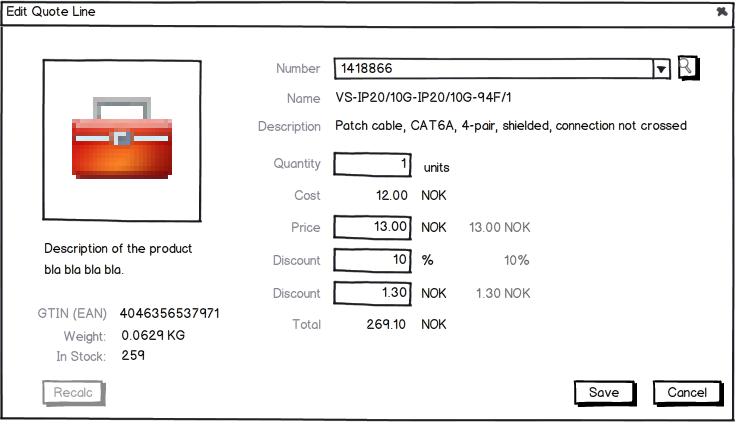


Figure 9: Please notice that this dialog has been revised severely. It will look very different in the final version.

The Quote Line dialog has five ways of setting the discount; using any of the following fields: DISCOUNT PERCENT and DISCOUNT AMOUNT, EARNING PERCENT and EARNING AMOUNT, and TOTAL PRICE.

These are linked. Change the discount percent, and the discount amount, the total price, the earning percent and earning amount will be recalculated.

We track which field was most recently modified by the user, and we use that as the master field. When UnitCostPrice, Quantity or UnitListPrice changes, the discounts can be adjusted accordingly to maintain the master field’s value.

e.g. Assume a quoteline where:

* UnitCost is 10.00
* UnitListPrice = 13.00

The user has entered

* Quantity = 10
* Discount 10%

The UserValueOverride is DiscountPercent since that is what the user modified last.

|  |  |
| --- | --- |
| The connector calculates the **SubTotal** from the Quantity and UnitListPrice: | 10\*13.00 = 130.00 |
| The connector calculates the **Discount** **Amount** from the Discount Percent and SubTotal: | 130.00 \* 10% = 13.00 |
| The connector calculates the **Total Price** from the SubTotal and Discount Amount: | 130.00 – 13.00 = 117.00 |
| The connector calculates the **Total Cost** from the Quantity and UnitCost: | 10\*10.00 = 100.00 |
| The connector calculates the **Earning Amount** from the TotalPrice and cost: | 117.00– 100.00 = 17.00 |
| The connector calculates the **Earning Percent** from the Earning Amount and TotalPrice: | 17.00 / 117.00 = 14.53% |

The user changes **Quantity**: Quantity = **100**

|  |  |
| --- | --- |
| The connector calculates the **SubTotal** from the Quantity and UnitListPrice: | 100\*13.00 = 1300.00 |
| The connector calculates the **Discount** **Amount** from the Discount Percent and SubTotal: | 1300.00 \* 10% = 130.00 |
| The connector calculates the **Total Price** from the SubTotal and Discount Amount: | 1300.00 – 130.00 = 1170.00 |
| The connector calculates the **Total Cost** from the Quantity and UnitCost: | 100\*10.00 = 1000.00 |
| The connector calculates the **Earning Amount** from the TotalPrice and cost: | 1170.00– 1000.00 = 170.00 |
| The connector calculates the **Earning Percent** from the Earning Amount and TotalPrice: | 170.00 / 1170.00 = 14.53% |

The user changes **Discount Amount** to 100 – this changes the UserValueOverride to DiscountAmount.

|  |  |
| --- | --- |
| The connector calculates the SubTotal from the Quantity and UnitListPrice: | 100\*13.00 = 1300.00 |
| The connector calculates the **Discount** **Percent** from the Discount Amount and SubTotal: | 100 / 1300.00 = 7.70% |
| The connector calculates the Total Pricefrom the SubTotal and Discount Amount: | 1300.00 – 100.00 = 1200.00 |
| The connector calculates the Total Costfrom the Quantity and UnitCost: | 100\*10.00 = 1000.00 |
| The connector calculates the Earning Amount from the TotalPrice and cost: | 1200.00– 1000.00 = 200.00 |
| The connector calculates the Earning Percent from the Earning Amount and TotalPrice: | 200.00 / 1200.00 = 16.67% |

The user changes **Total Price** to 1100 – this changes the UserValueOverride to TotalPrice.

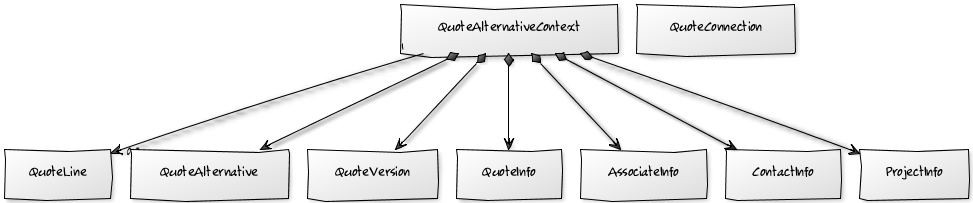
|  |  |
| --- | --- |
| The connector calculates the SubTotal from the Quantity and UnitListPrice: | 100\*13.00 = 1300.00 |
| The connector calculates the Discount Amount from the TotalPrice and SubTotal: | 1100 - 1300.00 = 200.00 |
| The connector calculates the Discount Percent from the Discount Amount and SubTotal: | 200 / 1300.00 = 15.40% |
| The connector calculates the Total Costfrom the Quantity and UnitCost: | 100\*10.00 = 1000.00 |
| The connector calculates the Earning Amount from the TotalPrice and cost: | 1200.00– 1100.00 = 100.00 |
| The connector calculates the Earning Percent from the Earning Amount and TotalPrice: | 100.00 / 1100.00 = 9.10% |

## QuoteAlternativeWithLinesInfo

Combines an alternative with the corresponding quote lines.

|  |  |
| --- | --- |
| QuoteAlternativeInfo CRMAlternative | Read + Write alternative information. |
| QuoteLineInfo[] CRMQuoteLines | The quotelines that the alternative consists of. |

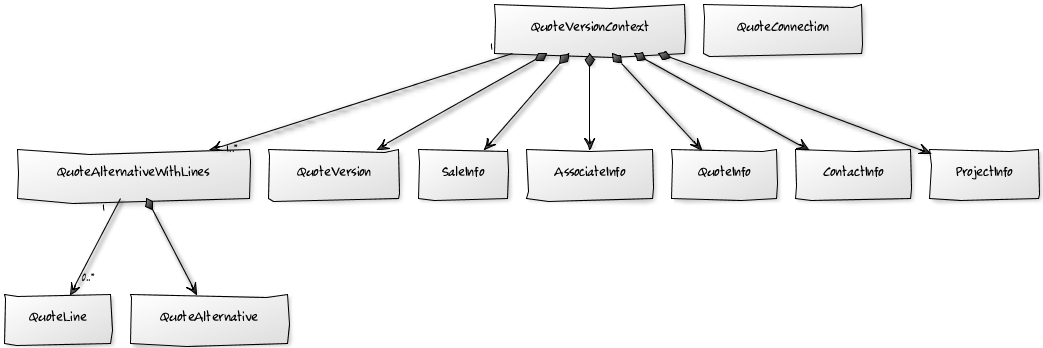
## QuoteAlternativeContextInfo #9545



Exists to be able to give the connector the relevant information for QuoteAlternative specific operations, for instance to be able to compute the correct price/discount on all levels.

|  |  |
| --- | --- |
| string ERPClientKey | Foreign key – always present. |
|  |  |
| UserInfo CRMAssociate | Read-only. The logged in user (the salesman) Note: not necessarily the the owner of the sale – that is exposed via the SaleInfo object |
| IContactInfo CRMCompany | Read-only. The sale’s customer info – more detailed information than is provided by the ISaleInfo object. |
| IPersonInfo CRMPerson | Read-only. The sales related person info (if any). |
| IProjectInfo CRMProject | Read-only. The sales related project info (if any). |
| ISaleInfo CRMSale | Read-only information about the sale the quote is attached to. The CRM client will update the amount + cost fields on the sale based on the quote values. |
| ForeignKeyInfo[] ForeignKeys | The foreign keys that is related to this quote.  Contact keys, project keys, sales keys and quotekeys. |
| QuoteInfo CRMQuote | Read + Write quote information |
| QuoteVersionInfo CRMQuoteVersion | Read + Write version information.  This is the active version |
| int EISConnectionId | The id of the EIS Connection in the CRM system. |
| string EISConnectionGuid | A conversation identifier |
| string ERPCompanyKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPPersonKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPProjectKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPSaleKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
|  |  |
|  |  |
| String UserLanguageCode | The language the users uses in the CRM client. The connector should preferably respond using this language. .net culture code: “nb-NO”, “en-US” etc. |
|  |  |
| QuoteAlternativeWithLinesInfo CRMAlternativeWithLines | Read + Write alternative information.  This is the active alternative |

## QuoteVersionContextInfo



Exists to be able to give the connector relevant information for QuoteVersion specific operations, like SendQuoteVersion.

|  |  |
| --- | --- |
| string ERPClientKey | Foreign key – always present. |
|  |  |
| UserInfo CRMAssociate | Read-only. The logged in user (the salesman) Note: not necessarily the the owner of the sale – that is exposed via the SaleInfo object |
| ISaleInfo CRMSale | Read-only information about the sale the quote is attached to. The CRM client will update the amount + cost fields on the sale based on the quote values. |
| IContactInfo CRMCompany | Read-only. The sale’s customer info – more detailed information than is provided by the ISaleInfo object. |
| IProjectInfo CRMProject | Read-only. The sales related project info (if any). |
| ForeignKeyInfo[] ForeignKeys | The foreign keys that is related to this quote.  Contact keys, project keys, sales keys and quotekeys. |
| int EISConnectionId | The id of the EIS Connection in the CRM system. |
| string EISConnectionGuid | A conversation identifier |
| string ERPCompanyKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPPersonKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPProjectKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
| string ERPSaleKey | A key to the equivalent id in the ERP system, from the EIS System.  May be empty. |
|  |  |
| QuoteInfo CRMQuote | Read + Write quote information |
| QuoteVersionInfo CRMQuoteVersion | Read + Write version information.  This is the active version |
|  |  |
| String UserLanguageCode | The language the users uses in the CRM client. The connector should preferably respond using this language. .net culture code: “nb-NO”, “en-US” etc. |
|  |  |
| QuoteAlternativeWithLinesInfo CRMQuoteAlternativeWithLines | Read + Write alternative information.  This is the active alternative |

## PriceListInfo #9547



A pricelist is basically a collection of products. It can be valid in a time period, and outright deactivated. All prices in the product list is in a specific currency.

We have decided not to separate prices and products, which means that we get a simpler data model, but some redundancy.

|  |  |
| --- | --- |
| string ERPPricelistKey | Reference to the pricelist in the product supplier system. |
|  |  |
| int QuoteConnectionId | The connection in SuperOffice this pricelist comes from. |
|  |  |
| string Name | Name of this pricelist to use in the user interface. |
| string Description | Description of this pricelist , will be used as tool-tip in the user interface. |
| string Currency | The ISO currency code, like 'USD' or 'NOK'. |
| string CurrencyName | The name to use in the user interface, like perhaps 'US dollar' or '$' |
|  |  |
| DateTime ValidFrom | The date (inclusive) the pricelist start to be valid. This can be DateTime.MinValue to signal that it doesn't have a specific start date. |
| DateTime ValidTo | The date (inclusive) the pricelist ends to be valid. This can be DateTime.MaxValue to signal that it doesn't have a specific end date. |
|  |  |
| Bool IsActive | Is the list active (as opposed to being worked on, suddenly canceled, etc. |

## ProductInfo (Article) #9548

A product is some thing or service that can be sold or leased to a customer.

The fields that will be copied to the QuoteLines are marked in grey.

|  |  |
| --- | --- |
| string ERPProductKey | Reference/foreign key to the product in the product supplier system, if it exists there. |
| string ERPPricelistKey | Foreign key to the price list that this product is a part of. |
| Bool InAssortement | True for products that should currently be offered, false when the product is discontinued and should not ordinarily be offered.  When false the product no longer appears in search results. |
| Decimal InStock | Negative numbers will be interpreted as how many are ordered. Might not be available. Requires the “Provide-Stock-data” capability, and that the ERP system is available. |
|  |  |
| string Name | The name to use in the user interface |
| string Description | The description to use, with potentially several lines.  Will be used as tool-tip to use in the list user interface too. |
| string Code | The product code / article number in the product supplier system. |
| string QuantityUnit | What is the unit (meter, ton, bushel, microsecond, gradus, τρυβλίον, 五合枡, دونم or whatever); Connector handles conversion relative to PriceUnit if they are different. |
| string PriceUnit | What is the unit (meter, ton, bushel, microsecond, gradus, τρυβλίον, 五合枡, دونم or whatever); read-only for lines that originate in defined products. |
| string ItemNumber | Line item number, NOR: «Postnummer». Specific numbers from some hierarchy, for instance “1.4.3.2P”. Typically used to sort the items in the quote by some standard way. |
|  |  |
| string Url | URL to product information web page |
|  |  |
| string ERPProductCategoryKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text with the actual product category. |
| string ERPProductFamilyKey | Either a List id to an id from a connector provided list, or, if the connection doesn't support lists, a text with the actual product family. |
| string ERPProductTypeKey | Either a List id to an id from a connector provided list, or, if the connection doesn’t support lists, a text with the actual product type. |
|  |  |
| string Supplier | Name of the supplier of the product |
| string SupplierCode | Suppliers part code/number or other key-like field |
|  |  |
| string Thumbnail | The thumbnail of the product, if it exists. Base64 encoded string, or a valid URI that resolves to an image.  Requires the “Provide-Thumbnail” capability. |
| string VATInfo | A field for putting VATInfo you need to show in the final quote document, like the VAT type that is used.  Not used in any business logic in SuperOffice; available to document templates. |
| string VAT | Tax/VAT if available from ERP system.  Could be either the percentage or the actual value.  This is just to help out the layout of the quote in a document, but SuperOffice will not try to calculate this value. |
|  |  |
| Decimal UnitCost | The cost price.  Might not be given, use Decimal.MinValue to signal this. |
| Decimal UnitMinimumPrice | The minimum price this salesman can offer to his customer. This might be cost price if there is no policy.  Might not be given, use Decimal.MinValue to signal this. |
| decimal UnitListPrice | (Basic Price, normal price, standard price.)  This is the basic price from which the discount is computed from.  The ListPrice will stay the same even when a larger amount is ordered. |
|  |  |
| ProductExtraDataFieldInfo[] ExtraData | Extra data (fields with labels). Shall be shown in the quoteline dialog.  Additional info that the ERP system would like to store and show in the user interface.  Information placed here is shown in the GUI if the “provide-extra-data” capability is true.  Different products can have different fields.  It will not be possible to directly put info here into the quote document.  BTW, this will be stored in the SuperOffice database as an xml field, like this:  <Fields>  <Field Name="Weight" Type="String"><![CDATA[[F:16.6] tons]]></Field>  <Field Name="Height" Type="String"><![CDATA[ [F:44.0]cm]]></Field>  <Field Name="Arms" Type="String"><![CDATA[ [I:2]]]></Field>  <Field Name="Certification" Type="String"><![CDATA[AB-ICE]]></Field>  <Field Name="Weight" Type="String"><![CDATA40°C]]></Field>  <Field Name="Security info" Type="Url" ><![CDATA[http://www.armystudyguide.com/content/army\_board\_study\_guide\_topics/hand\_grenades/throwing-of-hand-grenades.shtml]]></Field>  <Field Name="Security image ><![CDATA[http://upload.wikimedia.org/wikipedia/commons/thumb/8/80/MK2\_grenade\_DoD.jpg/220px-MK2\_grenade\_DoD.jpg</Field]]>  </Fields> |
|  |  |
| string Rights | Field1=right&Field2=right, etc. of any fields that have non-standard field access rights.  Rights can be one of: N (=None or Hidden), R (=Read-only), W (=Writeable), M (=Mandatory).  The fields will mostly be from the Quoteline table, but some added fields that are conceptually part of the quoteline, like Image will also be possibly to set rights on.  See Rights field for more information  Will be used by SuperOffice to control the user interface when showing the record. |
| string Rule | The names of one or more calculation rules that are in effect for this line, comma-separated case-insensitive.  Will NOT be used by SuperOffice. |
| string ExtraField1 | This a simple field for adding information that the Connector can provide, and that the quote document need to display. |
| string ExtraField2 | This a simple field for adding information that the Connector can provide, and that the quote document need to display. |
| string ExtraField3 | This a simple field for adding information that the Connector can provide, and that the quote document need to display. |
| string ExtraField4 | This a simple field for adding information that the Connector can provide, and that the quote document need to display. |
| string ExtraField5 | This a simple field for adding information that the Connector can provide, and that the quote document need to display. |

## ProductExtraDataFieldInfo

A way to show some simple extra data on a product, typically to help the user to identify the correct product. Basically a bucket of additional info that the ERP system would like to store and show in the user interface. Information placed here is shown in the GUI if the “provide-extra-data” capability is true.

|  |  |
| --- | --- |
| string FieldName | Label for the field |
| string FieldValue | Value for the field |
| ExtraDataFieldType Type | String, image, url.  How the value should be interpreted. |

To ensure that values in the value field is correctly displayed according to the user’s culture setup, we have a little system for making this work correctly. You just wrap the values in [] brackets with a format specifier, like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Valuetype | Symbol | Comment | Example |
| Date | D | Use [D:mm/dd/yyyy] | [D:01/01/2009] |
| DateTime | DT | Use [DT:MM/DD/YYYY 00:00:00.0000000] | [DT: |
| double | F | Use ‘.’ (period) as decimal separator. | [F:123.45] |
| Integer | I |  | [I:123] |
| Money | M | Use ‘.’ (period) as decimal separator. | [M:123.98] |

Which means that you can show several values in a field, like this: “*Between [D:12/01/2012] and [D:12/25/2012] it is a [F:99.5]% chance of meeting a Santa Clause.*” Which will translate into “*Between 01.12.2012 and 25.12.2012 it is a 99,5% chance of meeting a Santa Clause.*” with a Norwegian PC setup, for instance.

Or “Should be used in temperatures between [F:-30.0]°C and [F:50.0] °C.” -> “Should be used in temperatures between 30,0°C and 50,0 °C.”

## ExtraDataFieldTypeInfo

How should the ProductExtraDataFieldInfo value be interpreted?

* String
* Url
* Image (URL to image or Base64 encoded string)

## QuoteConnectorExtender

This class is made to make it easier to make some installation specific changes to a connector without changing the connector itself.

Just inherit QuoteConnectorExtender, override the function (-s) you need to change (or extend the functionality) and make SuperOffice connect to the new extender connector instead of the ERP connector (dynamicload section in the SuperOffcie.config file).

Fundamentally, this new connector puts itself between the ERPConnector and SuperOffice

## AddressInfo

|  |  |
| --- | --- |
| string AddressField1 | Address line |
| string AddressField2 | Address line |
| string AddressField3 | Address line |
| string AddressCity | The address city |
| string AddressZip | The address zipcode, typically a number. |
| string AddressZipGerman | Postcode for street address (for German addresses) |
| string CountryCode | The ISO country code, like, “NO” or “US”. See <http://www.iso.org/iso/home/standards/country_codes/iso-3166-1_decoding_table.htm#AA> for details. |
| string Country | The country name, as written on an envelope (“Norway”, “Sweden”, etc). |

# Use Cases

### Use Case – Adding a Product

The user will click the ADD button to open the QUOTE LINE DIALOG.

The user enters a search value: “Glops”

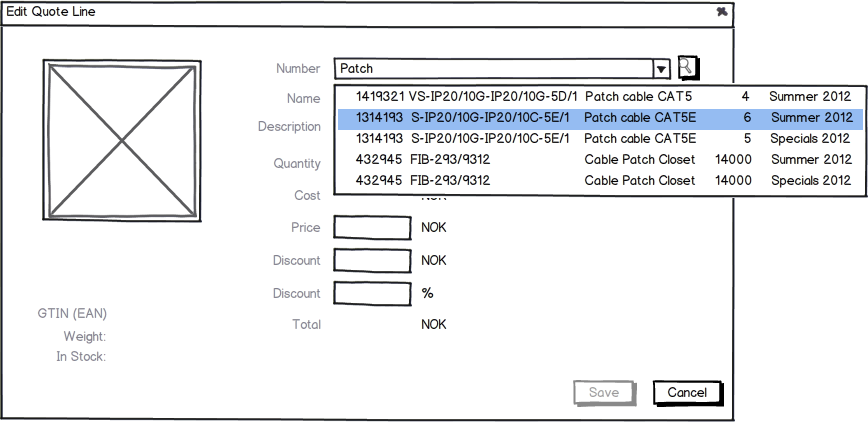


Figure 10: Please notice that this dialog has been revised severly. It will look very different in the final version.

The client calls the ERP connector **IProductProvider.FindProduct**(“Glops”).  
The ERP connector searches the ERP database and gets back a list of matching products.

The user selects a product from the result list. This gives us the ERP Product key of the product.

The client calls the **IProductProvider.GetQuoteLineFromProduct**( quoteContext, erpKey ) to get the full information for the product.

The client updates the dialog with values from the new QuoteLine.

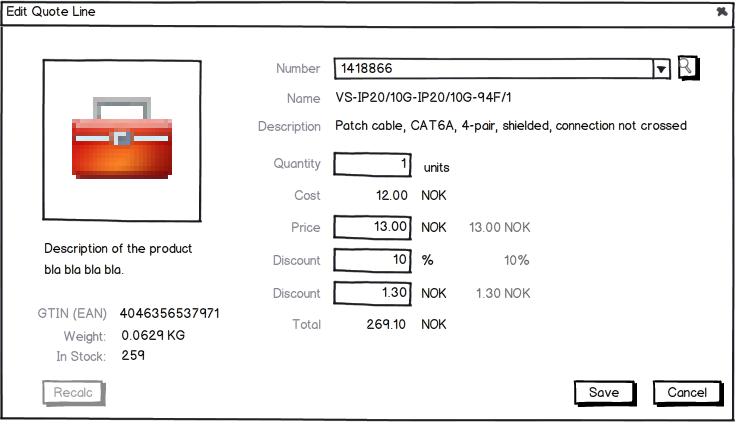


Figure 11: Please notice that this dialog has been revised severely. It will look very different in the final version.

*The user changes the QUANTITY from 1 to 10 and hits TAB*

The system calls ERP Connector: **IPriceProvider.OnQuoteLineChanged**( quoteline, quotecontext ) with the quoteline amount = 10 rather than 1. The ERP connector receives a new discount from the ERP system: Assume now that the ERP system thinks 10% would be a good discount for this quote.  
So the QuoteLine.ERPDiscountPercent = 10.

The system calculates the other value: ERPDiscountAmount = 10% \* UnitListPrice \* Quantity.

Since the UserValueOverride = None, this means that the ERP connector shall copy the ERPDiscountAmount into the DiscountAmount field and the ERPDiscountPercent in the DiscountPercent field.

Then the connector calculates the TotalPrice, the Earning percent and the Earning amount.

The system updates the GUI with the new values from the quote line.

*The user changes the UnitListprice from 13.00 to 14.00*

Again the system calls ERP Connector: **IPriceProvider.OnQuoteLineChanged**

Assume now that the ERP connector returns with the ERPDiscountPercent field updated to 12.

The system calculates the ERPDiscountAmount.

The system sees that the UserValueOverride is still none, so both ERP values are copied to the user discount fields.

Then the connector calculates the TotalPrice, the Earning percent and the Earning amount.

The system updates the GUI with the new values from the quote line.

*The user changes the DISCOUNT PERCENT from 12% back to 10%*

The system sets the Quoteline. UserValueOverride = DiscountPercent

Again the system calls ERP Connector: **IPriceProvider.OnQuoteLineChanged**

The system calculates the ERPDiscountAmount from the ERPDiscountPercent, but this time does not copy the ERP values to the user discount fields, since the user has entered a discount value (the UserValueOverride is not NONE).

The system calculates the QuoteLine.DiscountAmount from the DiscountPercent value since the UserValueOverride field is PercentField. And then the TotalPrice, the Earning percent and the Earning amount.

The user clicks the SAVE button.

The system calls the ERP Connector: **IPriceProvider.RecalculateAlternative**() to update all the quote alternative total.

### Use case – Dealing with customer specific product codes

Some, typically large, customers demand that you send quotes to them using the customers product codes. How should you go about solving this with SuperOffice?

There must be an alias database somewhere who knows how to translate the product codes into customer codes. We call this the “Alias database”.

When a user uses the fast search (that’s the “google” search in the QuoteLine dialog) he enters the proprietary customer alias “BOEING\_111”.

1. Your connector sends a search to the Alias database to see if the customer + search word (-s) have a match.
2. That search returns a match on the “BOEING\_111” which is “Pcx\_222”
3. You then add the word “Pcx\_222” to the user input
4. Pass the changed user input to the SuperOffice connector.
5. The SuperOffice connector then returns a match on “Pcx\_222”.
6. You then change the product code “Pcx\_222” in the result to the correct customer code “BOEING\_111”.
7. The user sees that he found the “BOEING\_111” product and accepts it.

Performance gains:

* We assume that only a few of the customers have this issue. Perhaps you should therefor consider to mark a quote with if it is such a customer and only search the alias database if the customer has actual aliases.
* You could easily create what we call a partner table in SuperOffice and fill it with the alias data. Such a table would contain the following columns: id, customerid, customer alias and your product code. It would automatically be transported onto travel, if you configure it to do so. All you have to do is, like with the product register, to synchronize it regularly.

When it comes to the complex search, where the user can specify which fields he want to search in, you can make your own search-provider. This provider could make it possible for you to allow the user to enter text into a search criteria you can call “Customer Code”. It would then be easy to perform the correct search against the alias data store (whether internally to SuperOffice or externally) and present the user with a result that shows the customer code or whatever you need.

# QuoteConnectorBase implementation

The QuoteConnectorBase implements most of the IQuoteConnector API, and adds some useful default behavior to the basic API contract. For example – recalculate alternative is handled for you.

In addition to the IQuoteConnector interface, the base class adds a few public methods of its own.

ValidateQuoteVersion calls the **ValidateVersion** implementation method, which calls down the hierarchy in order.

|  |  |
| --- | --- |
| QuoteVersionResponseInfo **ValidateVersion**(QuoteVersionContextInfo context, bool clearOldValues = false) | Validates the version, looks for problems. Will typically change the Status and Reason fields.  Should for instance validates the alternatives and then concatenates the problems into the Status and reason fields. |
| QuoteAlternativeWithLinesInfo **ValidateAlternative**(QuoteAlternativeWithLinesInfo quoteAlternativeWithLines, bool clearOldValues = false) | Check rules for the quote alternative and fill out the status and reason fields if there is one or more problems. |
| QuoteLineInfo **ValidateQuoteLine**(QuoteLineInfo ql, bool clearOldValues = false); | Check rules for the quoteline and fill out the status and reason fields if there is a problem. |

These functions are not part of the public API, but are part of the connector base class’s implementation.

You need to supply your own:

* InitializeConnection
* CanProvideCapability
* OnAfterSaveQuote
* OnBeforeDeleteQuote
* FindProduct
* GetProduct
* GetProducts
* GetQuoteLinersFromProduct

The SuperOffice connector is built on top of the QuoteConnectorBase, but it is tightly coupled to the SoDatabase and SoCore assemblies. This means that if you sub-class the SuperOffice connector, you will break whenever a new version is released.

IQuoteConnector

QuoteConnectorBase

SuperOffice QuoteConnector

# QuoteConnectorExtender implementation

The QuoteConnectorExtender implements the IQuoteConnector API by wrapping another connector, and delegating all calls to the wrapped connector.

To use the extender, you sub-class the QuoteConnectorExtender, and pass the name of the connector you would like to extend as part of the constructor call.

You then override any API calls you want to change, and leave the rest to the base implementation – which just forwards the calls to the wrapped connector.

IQuoteConnector

QuoteConnectorBase

SuperOffice QuoteConnector

QuoteConnectorExtender  
\_wrappedConnector

MyQuoteConnector  
: base( "SuperOfficeQuoteConnector")  
public override OnQuoteLineChanged(…)

Here we extend the SuperOffice quote connector, but we override the default implementation of OnQuoteLineChanged with our own custom logic.

MyQuoteConnector tells the extender to wrap the SuperOffice quote connector by passing the name to the base constructor.

Because the QuoteConnectorExtender class lives in the Plugins DLL, the MyQuoteConnector avoids taking direct dependency on the SoCore and SoDatabase DLLs, so it won’t be affected when a version change updates the SoCore assembly.

# How to create a SuperOffice Quote Connector

## Prerequisites

* Install Visual Studio
* Install SuperOffice Sales & Marketing Windows version on the computer. (The example assumes that you have installed the windows client on “C:\Program Files (x86)\SuperOffice\SuperOffice 7 Windows\”.)

## Basics

1. Open Visual Studio, but **Run as Administrator**. You will need this to be able to set the build output to a subfolder of Program files.
2. Select “Create new project” from the menu
   1. Select “Visual C#”
   2. Select “Class library” and give it a name, like “MyQuoteConnector” (or “SAP Connector”)
   3. Select OK
3. Right click on **References** in the Solution explorer and select “Add references…”
   1. Select “Browse”, and “Browse” again.
   2. Navigate to where you installed SuperOffice, typically “C:\Program Files (x86)\SuperOffice\SuperOffice 7 Windows” and select:
      1. SOCore.dll
      2. SuperOffice.Plugins.dll
4. Right-Click on the project in solution explorer and select “**Properties**”.
   1. Select “Build” tab
   2. Set “Output path” = C:\Program Files (x86)\SuperOffice\SuperOffice 7 Windows\
   3. Save & Close properties window.
5. Open “Class1.cs”
6. Insert “using SuperOffice.CRM;”
7. (If you are using the version that was made around Christmas 2012) Insert “using SuperOffice.Plugins.CRM.Sale;”
8. Insert “[QuoteConnector( Name )]” over the class def. This attribute is used to identify the class by SuperOffice as an ERPQuoteConnector.
9. Insert “**public const string Name = "MyQuoteConnector";**” inside the class. This name is used (via the attribute over) to identify the class by SuperOffice when it is dynamically loaded from the SuperOffice.config file. You shall have to insert this name and the name of the .DLL into the SuperOffice.config file’s DynamicLoad section later.

## If you just want to replace or extend part of an ERP connector

1. Right click on **References** in the Solution explorer and select “Add references…”
   1. Select “Browse”, and “Browse” again.
      1. Navigate to where you installed SuperOffice, typically “C:\Program Files (x86)\SuperOffice\SuperOffice 7 Windows” and select the connector you want to extend, for instance: SuperOffice.QuoteConnector.dll
2. Inherit “SuperOffice.CRM.**QuoteConnectorExtender**” in Class1
3. Create a constructor where you reference the ERP connector you want to extend, something like this:

public Class1()

: base(new SuperOffice.QuoteConnector.SuperOfficeQuoteConnector()) { }

1. Override the function(-s) you want to override. (for instance “OnQuoteLineChanged”)
2. Goto “When Done” ☺

## If you don’t want to extend an existing ERP connector

1. Inherit “SuperOffice.CRM.**QuoteConnectorBase**” in Class1. This class has a lot of what you want to do implemented already; it implements IQuoteConnector and IPlugin.
   1. Implement the abstract class SuperOffice.CRM.QuoteConnectorBase.
2. Implement **Quote Connector Setup functions**
3. Implement **Product Provider functions**
4. If you want to handle Orders, implement the Order Consumer functions
5. Goto “When Done” ☺

## When Done

1. Build your connector DLL
2. **DynamicLoad**Either change the config file or use the SOLOADER panel in Windows Admin client.
   1. Open “C:\Program Files (x86)\SuperOffice\SuperOffice 7 Windows\SuperOffice.config”
   2. Insert under:

<configuration>

<SuperOffice>

<Factory>

<DynamicLoad>

The folloing line to load your connector:

<add key="MyQuoteConnector" value="MyQuoteConnector.dll" />

1. Run admin
2. Click on the “Quote” element in the navigator on the left side.
3. Observe that you can see the Connector in the connector list
4. Add a connection to you connector.
5. Start SuperOffice win client
6. Create a new sale.
7. Create a quote with the new connector.

1. This is not working correctly in the 2013 June alpha release, but bear with us, it will soon. [↑](#footnote-ref-1)