

SYSPRO 8

Insight Tile Builder Reference

Technical Article

Last Published: January 2022



SYSPRO Help and Reference

Copyright © 2022 SYSPRO Ltd

All rights reserved

No part of this document may be copied, photocopied, or reproduced in any form or by any means without permission in writing from SYSPRO Ltd. SYSPRO is a trademark of SYSPRO Ltd. All other trademarks, service marks, products or services are trademarks or registered trademarks of their respective holders.


SYSPRO Ltd reserves the right to alter the contents of this document without prior notice. While every effort is made to ensure that the contents of this document are correct, no liability whatsoever will be accepted for any errors or omissions.

This document is a copyright work and is protected by local copyright, civil and criminal law and international treaty. This document further contains secret, confidential and proprietary information belonging to SYSPRO Ltd. It is disclosed solely for the purposes of it being used in the context of the licensed use of the SYSPRO Ltd computer software products to which it relates. Such copyright works and information may not be published, disseminated, broadcast, copied or used for any other purpose. This document and all portions thereof included, but without limitation, copyright, trade secret and other intellectual property rights subsisting therein and relating thereto, are and shall at all times remain the sole property of SYSPRO Ltd.

Contents

Introduction.....	5
Audience.....	5
Insight Tile Overview	6
Summary tiles.....	6
Insight Tile Detail Drilldown.....	7
Tile type - Text.....	9
Tile type - Chart	11
Insight Tile KPIs.....	13
Adding a Tile to a User Workspace.....	14
Adding a non-Context Tile to a User’s Menu.....	14
Adding a Context Tile to a User’s Application	19
Creating a Custom Tile with Context	23
Insight Tile Definition.....	23
New Tile Dialog.....	25
Insight Tile Builder - Tile Properties.....	26
Summary SQL	28
Detail SQL.....	34
Text Tile Preview.....	41
Save and Add to a Workspace.....	43
Insight Tile Customization	47
Text Tile Formatting.....	47
Text Tile Detail Formatting.....	52
Text Tile Detail – Column type override	59
Smart editing - AutoCurrency.....	63
Parameters	65
Parameters – Worked Example.....	65
Parameter types.....	75
Parameter used to override context key.....	78

Alternate tile definitions	79
Creating the tile definition	80
Adding a tile to user’s workspace	84
Viewing the tiles at run time.....	85
Insight Tile Properties – Business Activities	86
Data Sources and Business Views	88
Introducing Business Views.....	88
Advantages of Business Views	89
Creating a Business View – Worked example	90
Creating an Insight Tile using a Business View	93
Benefits of Insight Tiles using Business View Data Sources.....	98
Charts	99
Chart types: Bar and Line.....	99
Creating a Chart type Insight Tile.....	100
Creating a Line Chart to View Sales.....	110
Additional notes about chart tiles	114
Insight Tile Targets and KPIs	116
KPI Attributes	116
KPI Goal Types	116
KPI Threshold comparison – Value and KpiValue.....	119
Tile Icons and colors	119
Tile Colors.....	121
KPI Level	122
KPI Definition Access Control	122
Example: Standard tile – average days to pay for branch.....	123
KPI Permissions at the user level.....	127
Insight Tile Import and Export	130
Tile Export	130
Tile Import.....	133
Tile Import and Export Notes	135
Appendix 1 – List of Key Types	136



Appendix 2 – Data types	139
Default data types: Alphanumeric – ‘String’.....	139
Default data types: Numeric – ‘AutoNumber’	140
Default data types: Date – ‘DateLong’	140
Appendix 3 – System variables - \${Variable}	141
Appendix 4 – AutoNumber and AutoCurrency	142
Appendix 5 – System Limits	144



Introduction

Insight Tiles allow you to place relevant, timely and accurate business information on the user's desktop.

This can greatly aid their ability to make important business decisions and take corrective action all without leaving their SYSPRO environment.

This document will introduce **Insight Tiles** guiding you through the steps to apply existing **Insight Tiles** to a user's desktop and provide a deeper dive into how you can customize these Insight Tiles, or even create your own.

It is also possible to create a library of custom Insight Tiles on your local system and import them at an end-user's site. Therefore, the Import and Export capability will also be covered in this document.

AUDIENCE

This document is aimed at SYSPRO implementers, support personnel and SYSPRO system administrators. It is assumed that you have the necessary permissions when viewing or editing **Insight Tiles** and when applying them to a user's workspace.

A good knowledge of relevant SYSPRO database tables is useful as is a good knowledge of SQL Server Transact-SQL syntax, however by combining **Insight Tiles** and the **Business Activity Query (View) Builder**, you can greatly simplify the requirements to understand the underlying SYSPRO database tables and SQL statements required when building an **Insight Tile**.

Insight Tile Overview

Firstly, it's important to be clear on exactly what do we mean by an **Insight Tile** in SYSPRO.

An **Insight Tile** is a small square or rectangular area on your workspace (an area within your SYSPRO application) that can either contain purely text or a chart. These are known as 'text' and 'chart' tiles respectively.

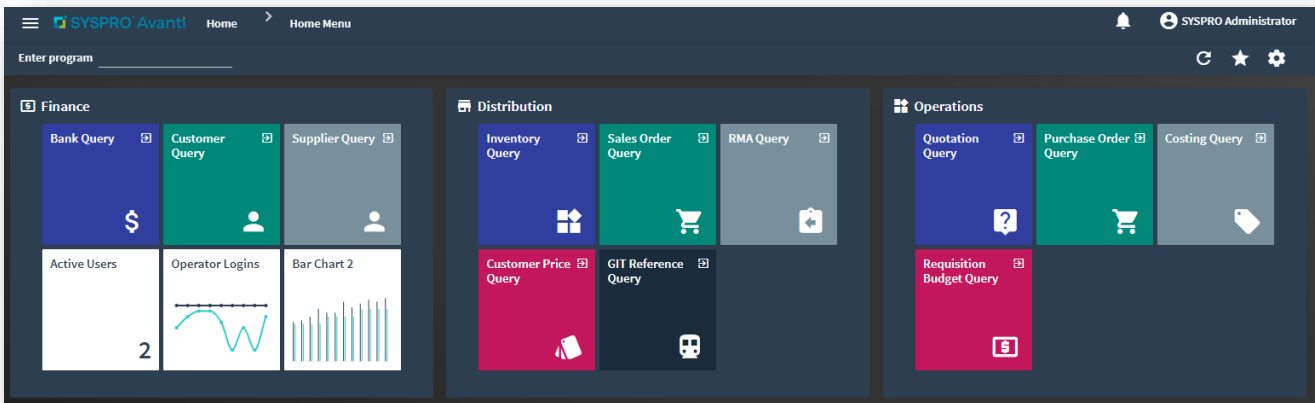
The tiles visible on a user's workspace are known as **summary tiles** as they show summarized or aggregated information.

SUMMARY TILES

Summary tiles are further categorized into **Insight Tiles** that can be placed anywhere (such as on the menu) and do not need any specific context apart from the current SYSPRO environment, and **Insight Tiles** that are designed to be placed within a SYSPRO application and require a specific context key, such as the current Customer, Stock Code, Sales order, Job etc.

SUMMARY TILES – NON-CONTEXT

In the example below, we have customized the user's menu and added three tiles with a white background to the Finance pane. These tiles do not require any specific context keys to be passed.

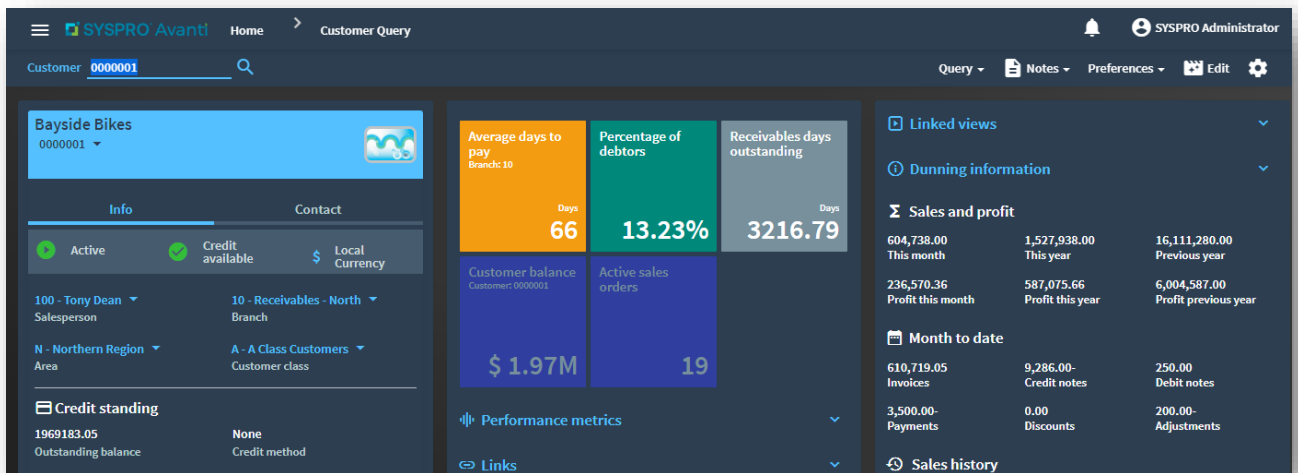


These tiles consist of:

- A text tile titled 'Active Uses' that shows the number of users logged into SYSPRO on this site (in this example we are using a test system)
- A line chart tile titled 'Operator Logins' that compares the active users with the licensed users for the recent past (again this is a test system and shown as an example)
- A bar chart tile titled 'Bar Chart 2' that is just a dummy chart to demonstrate the bar chart capabilities and does not perform any business function.

SUMMARY TILES – CONTEXT DRIVEN

In the example below, we have customized the user's Customer Query workspace and added five text **Insight Tiles** to the central pane. Each of these tiles use the current customer key as the context for the information shown.



Note: As the user selects a different customer, the tiles will be refreshed with information relevant to the current customer key.

INSIGHT TILE DETAIL DRILLDOWN

For text-type **Insight Tiles**, as shown in the customer example above, it can be useful to drilldown into the detail that makes up the tile summary information.

For example, if the user clicks on the tile that shows that there are 19 Active sales orders:



The system will drilldown to the detail and display the list of all 19 Active sales orders:

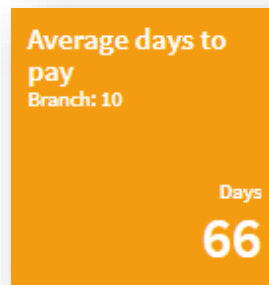
Active sales orders

Customer name	Salesperson	Branch	Sales order	Order status	Customer purchase order number
Bayside Bikes	100	10	000792	4	BB781
Bayside Bikes	100	10	000828	4	BB 810
Bayside Bikes	100	10	000834	4	BB 815
Bayside Bikes	100	10	000836	1	BB 7854
Bayside Bikes	100	10	000839	8	N900805-2005
Bayside Bikes	100	10	000841	8	BB 810
Bayside Bikes	100	10	000845	8	BB 900
Bayside Bikes	100	10	000847	F	BB 4789
Bayside Bikes	100	10	000851	1	212111
Bayside Bikes	100	10	000853	1	BB 901
Bayside Bikes	100	10	000854	1	1211Q
Bayside Bikes	100	10	000855	1	212232F
Bayside Bikes	100	10	000858	1	2111
Bayside Bikes	100	10	000859	1	4567

Showing 19 item(s)

This allows you to see the detail that makes up the summary value, and to further drilldown by clicking on the hyperlinks provided, for example to see the salesperson's details or to view more information about each specific sales order.

Similarly, if you click on the Average days to pay summary tile:



The drilldown will display the following detail:

Average days to pay

Customer	Name	Salesperson	Average Days to Pay	Customer Branch	Age of zero bal inv
0000001	Bayside Bikes	100	75.00	10	7
0000002	Bikes & Blades - North	100	51.00	10	15
0000008	Garden and Sports	102	66.00	10	6
0000010	Maniac Sports - North	100	100.00	10	10

Showing 4 item(s)

Note: Drilldown detail is only available for 'text' type tiles. The drilldown capability is not relevant for chart tiles.

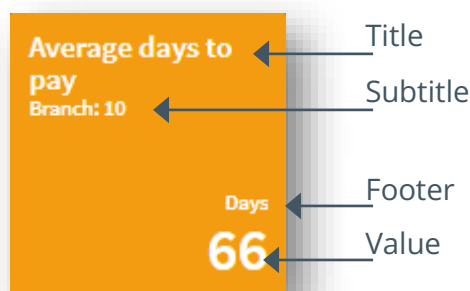
When customizing an **Insight Tile**, you can change everything that is shown, both on the text tile, the chart tile and the columns shown when using the detail drilldown.

TILE TYPE - TEXT

The most common and simplest to define **Insight Tiles** are 'Text' tiles.

Text tiles allow you to show a simple or aggregated value or a text string highlighting an attribute that is important to the user.

A text tile consists of up to four components as shown below:



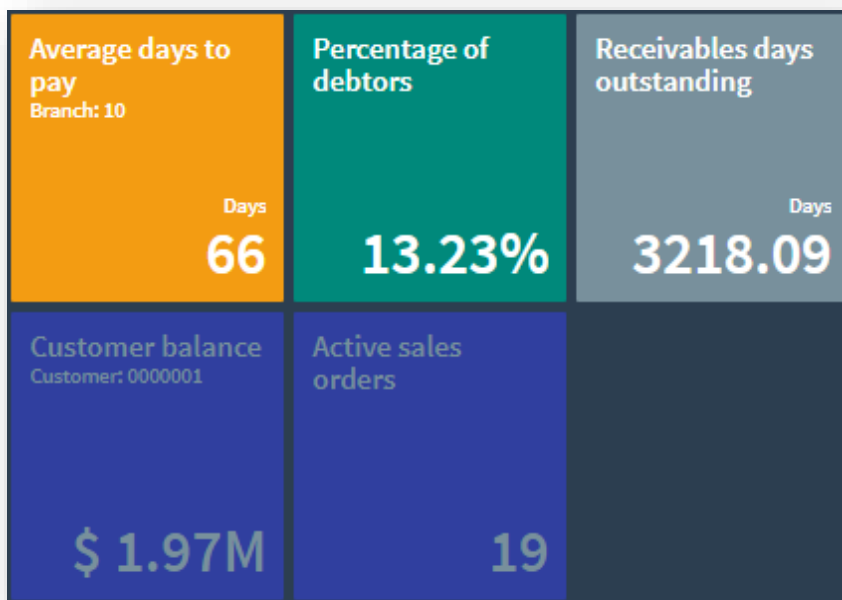
Note that the size, alignment, and font of each text component may vary depending on your SYSPRO release and theme selected. The colors can be defined when the tile is added to the user's workspace and modified based on a KPI threshold at run time.

Also note that in this example the Footer field is above the Value field.

The following table describes the text tile component attributes:

Component	Typical font size	Mandatory or Optional	SQL Column (exact case)
Title	Medium	Mandatory	{not applicable}
Subtitle	Smallest	Optional	SubTitle
Value	Largest	Mandatory	Value
Footer	Smallest	Optional	Footer

The following screenshot shows a group of text tiles, each with various components:



Taking the top left tile and moving across to the right and then bottom left across to the right, the tiles have the following components:

- Title, Subtitle, Value, Footer (all components)
- Title and Value
- Title, Value and Footer
- Title, Subtitle and Value
- Title and Value

As mentioned, the tile Value component is the key reason for the tile and is usually a numeric value – either directly from the database, calculated or aggregated in some way.

However, it is also possible for the Value component to be an alphanumeric string with any relevant characters. For example, it could be a status or location.

Text tiles are relatively small, therefore it's important to show concise information to retain readability and effectiveness.

TILE TYPE - CHART

Insight Tiles can be defined as a 'Chart' type.

Chart tiles allow you to graph a set of data representing a trend and/or comparing values.

There are currently two chart types:

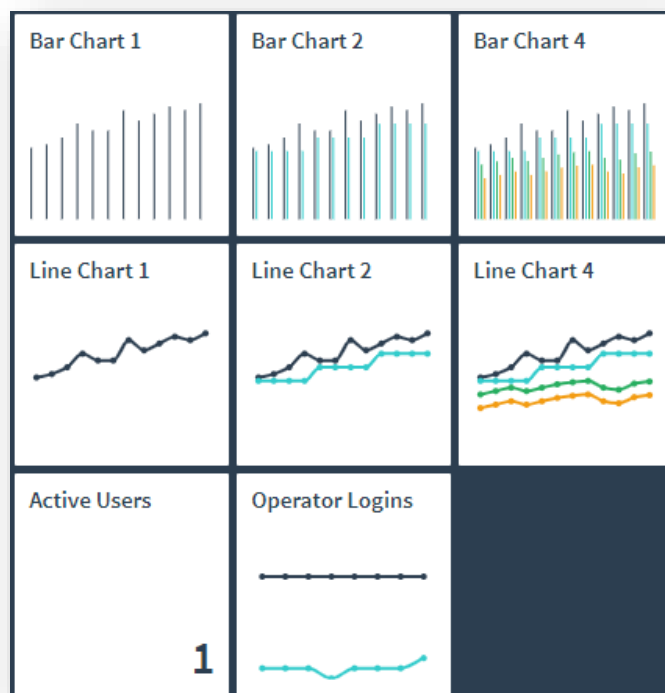
- Line
- Bar

Each chart type supports up to 4 series, each up to 24 points.

Remember that chart tiles are physically small, so the overall trend is usually the most important indicator when showing a chart. The user can only see the actual values by moving the mouse over the chart – the tooltip will show the series and item value.

You can select the small or large tile size when adding a tile to the user's workspace.

The following screenshot shows several chart tiles – most of these tiles were created for demonstration purposes only.



A chart tile consists of several components:

- Title
- Mandatory Series 1 set of up to 24 values
- Optional Series 2 set of up to 24 values
- Optional Series 3 set of up to 24 values
- Optional Series 4 set of up to 24 values

Each series value can have a tooltip.

Note that the size, alignment, and font of the title component may vary depending on your SYSPRO release and theme selected. The colors can be defined when the tile is added to the user's workspace.

The following table describes the chart tile component attributes:

Component	Mandatory or Optional	SQL Column point value	SQL Column Y-Axis label (*)	SQL Column Tooltip value
Title	Mandatory	{Not applicable}	{Not Applicable}	{Not applicable}
Series 1	Mandatory	P1 thru P24	Lab1 thru Lab 24	Tip 1 thru Tip24
Series 2	Optional	P1 thru P24	Lab1 thru Lab 24	Tip 1 thru Tip24
Series 3	Optional	P1 thru P24	Lab1 thru Lab 24	Tip 1 thru Tip24
Series 4	Optional	P1 thru P24	Lab1 thru Lab 24	Tip 1 thru Tip24

The SQL statement is expected to return up to 24 data points labelled P1 thru P24. For example, if you only want to return, say, 12 points then your SQL statement should return values for columns P1 thru P12.

In addition, you can return a string to be appended to the tooltip value – typically indicating the Y-Axis caption but could contain any string.

(*) Y-Axis label values named Lab1 thru Lab24, are not currently shown when rendering the chart but must be supplied for the data point to be shown.



INSIGHT TILE KPIS

Showing relevant and timeous context-based information to a user can be very useful, however additional business value can be derived by highlighting when a tile value exceeds a specific threshold or value.

SYSPRO **Insight Tiles** allow you to assign warning and critical thresholds for each text tile. This allows you to change the background and foreground colors and override the icon.

A system administrator can define KPI thresholds on a system-wide, company, role, or user basis. Permissions to edit KPIs are granted on a per-operator basis.

A supervisor can set KPI thresholds for a specific role so that everyone assigned to the role has the same set of KPIs.

If required, you can allow a user to define their own 'user' KPIs. This can be useful when the organization allows users to be self-managing.

For more information see the topic: [Insight Tile Targets and KPIs](#).

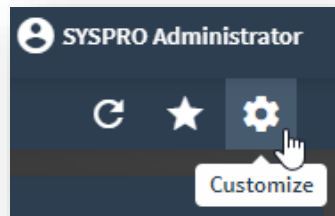
Adding a Tile to a User Workspace

When showing examples of how to add tiles to a user's workspace and viewing the completed tiles at run time we will be using the SYSPRO Web-UI (Avanti), however it is also possible to add **Insight Tiles** to the SYSPRO Rich client interface.

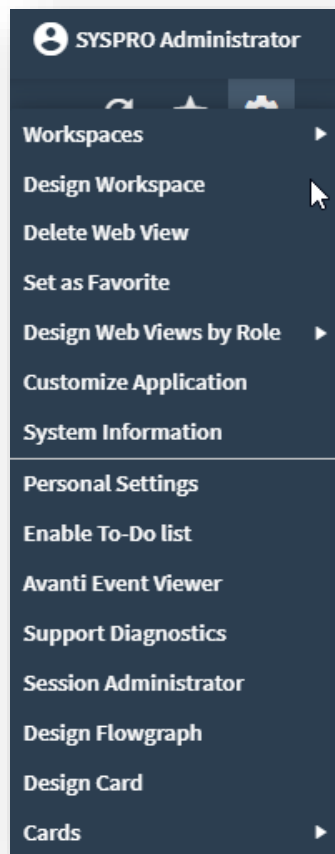
This topic is not designed to be an exhaustive explanation of editing a user's workspace but will contain the key points relevant when adding an **Insight Tile**. In all cases the options available may vary depending on your environment and the exact SYSPRO version installed on your site. The main steps will still be relevant.

ADDING A NON-CONTEXT TILE TO A USER'S MENU

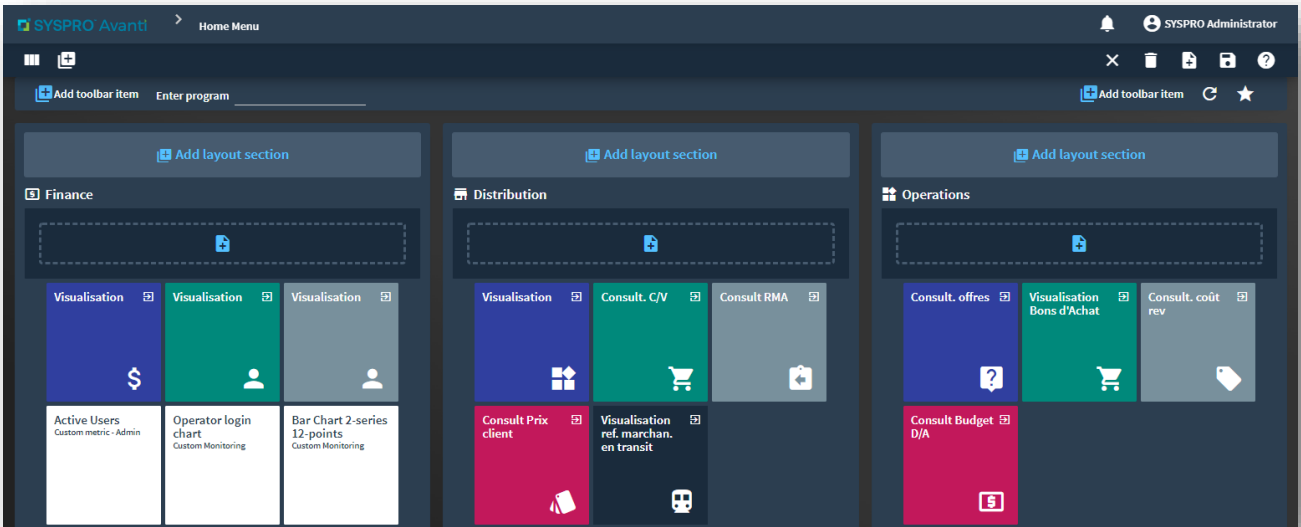
Login to SYSPRO Avanti and select the 'Customize' icon (top right):



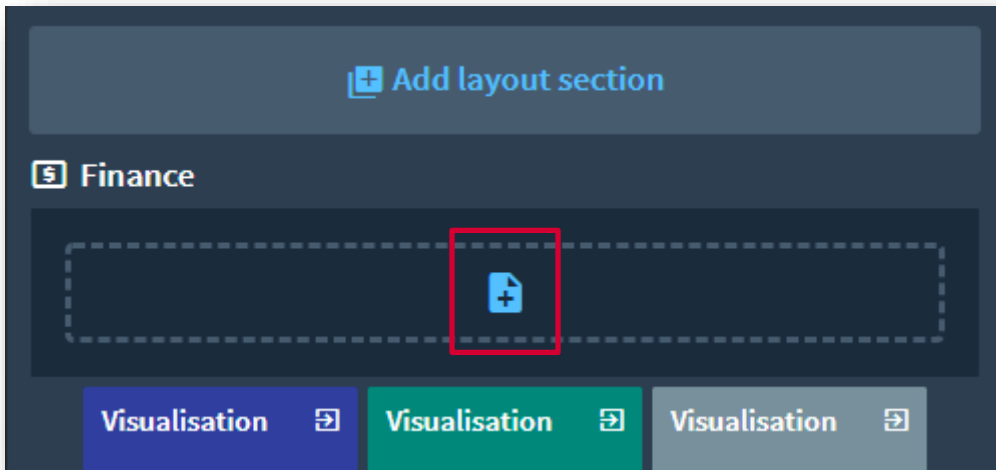
This will show a dropdown list of items – select **Design Workspace**:



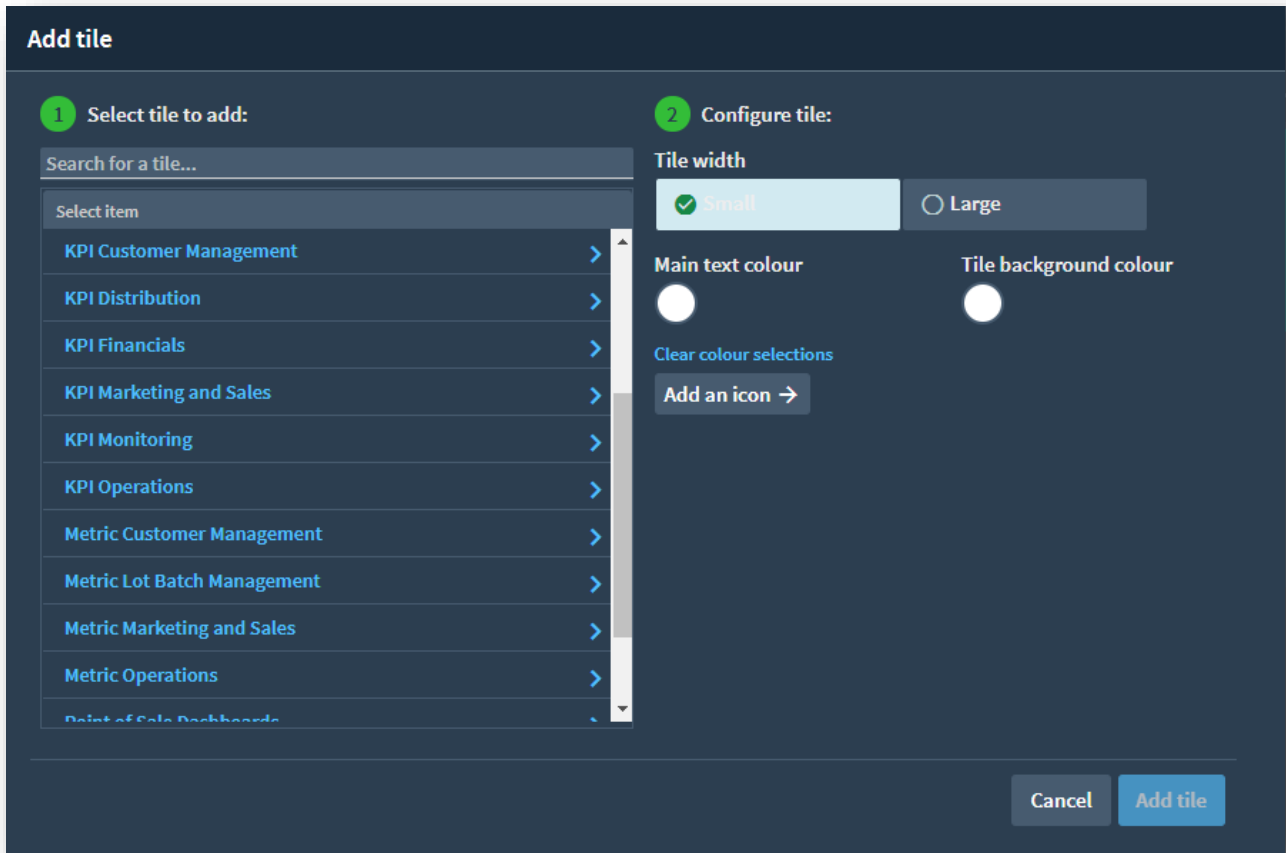
After a few seconds, the workspace editor is displayed. The editor provides a visualization showing how the user will experience this workspace.



To add a new tile, click on the '+' symbol in the layout section:



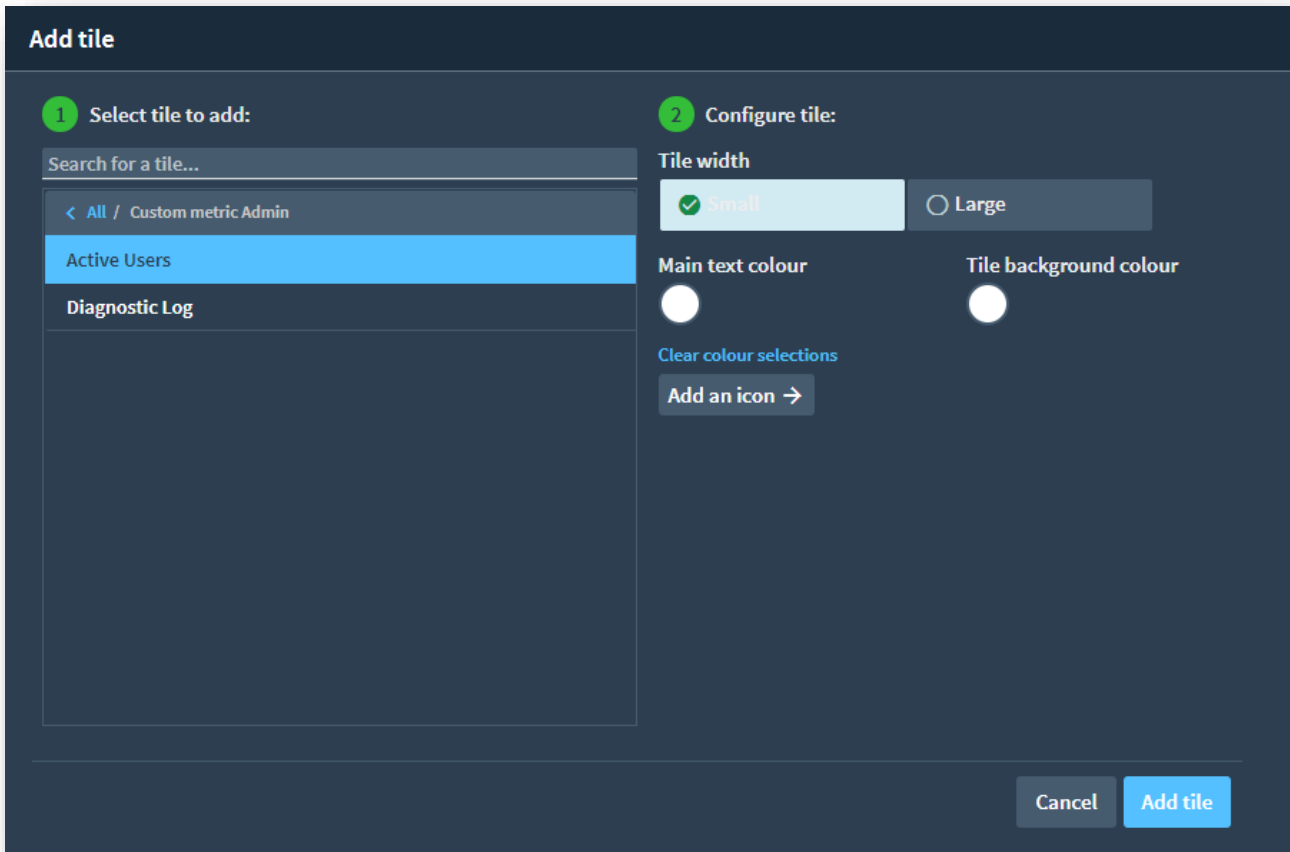
You can then locate the **Insight Tile** that you wish to add:



Tiles are shown grouped by Category to make it easier to find relevant tiles. You can also type in the name (or a partial string) against 'Search for a tile...' to find tile descriptions that match the text entered.

In this example, we are adding a tile to the user's menu, so we should select a tile that does not require a context key to be passed.

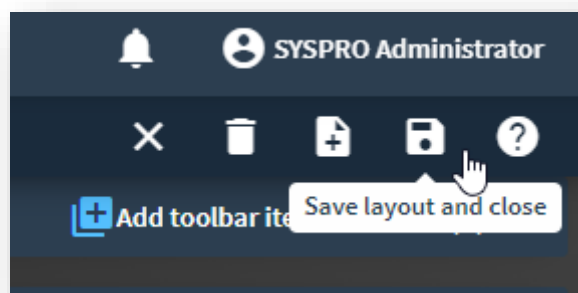
We are going to select a custom **Insight Tile**. We have selected the category **Custom metric Admin** and the tile 'Active Users'.



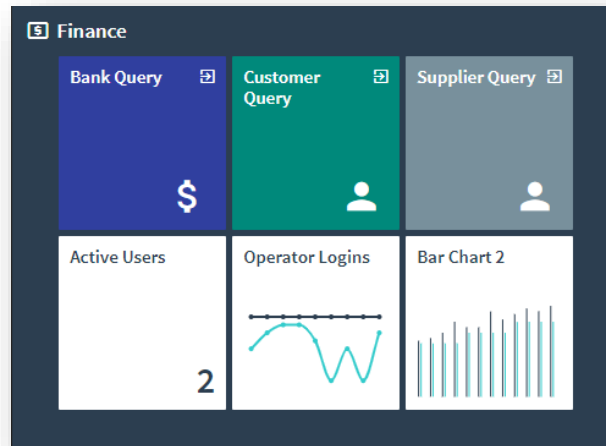
You can configure the tile by selecting the tile size (Small or Large), the tile Main text color and Tile background color and add an icon if required.

In my case I already have several **Insight Tiles** on the workspace, so I have moved the newly selected tile to the beginning of my tiles.

We can now save the layout and close the customization – click on the Save icon (top right of the menu bar).

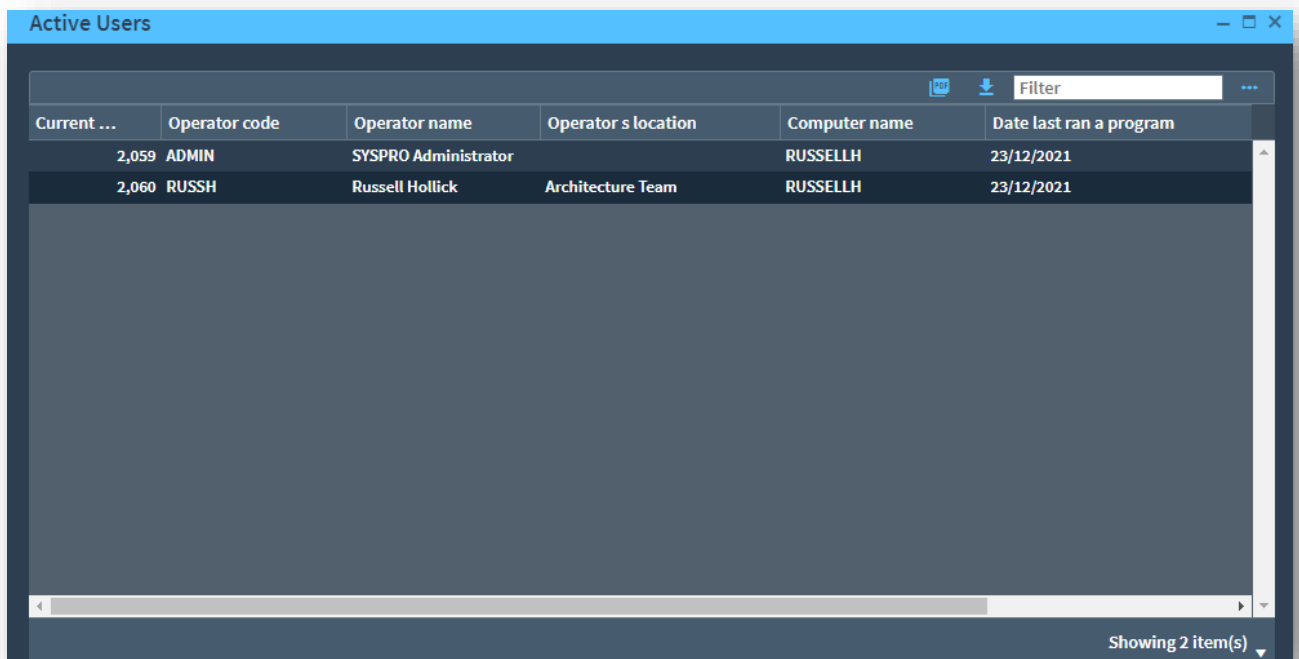


The menu will be refreshed with the new layout. Each **Insight Tile** is also refreshed.



Note the new 'Active Users' tile showing that there are 2 users logged into SYSPRO (I placed it at the bottom left of the existing tiles).

You can click on the **Insight Tile** to drilldown into the list of users running SYSPRO.

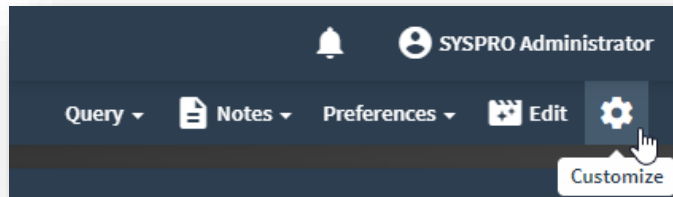


Note: This is a test system. I have logged into SYSPRO with both a user named ADMIN and user named RUSHH.

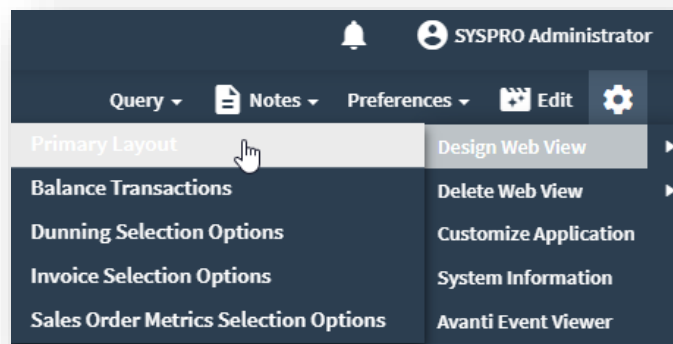
ADDING A CONTEXT TILE TO A USER'S APPLICATION

In this example we're going to add an **Insight Tile** to the Customer Query application. The current customer code will be passed to the tile so that information relating to the current customer will be shown.

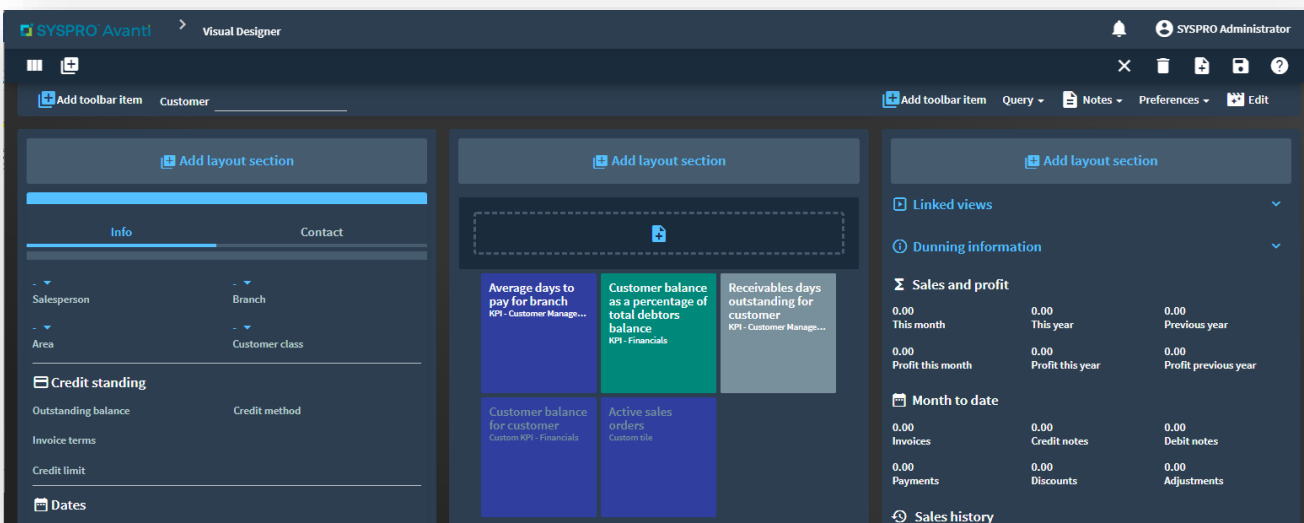
Login to SYSPRO Avanti and select the Customer Query application. Once loaded, click the 'Customize' icon (top right):



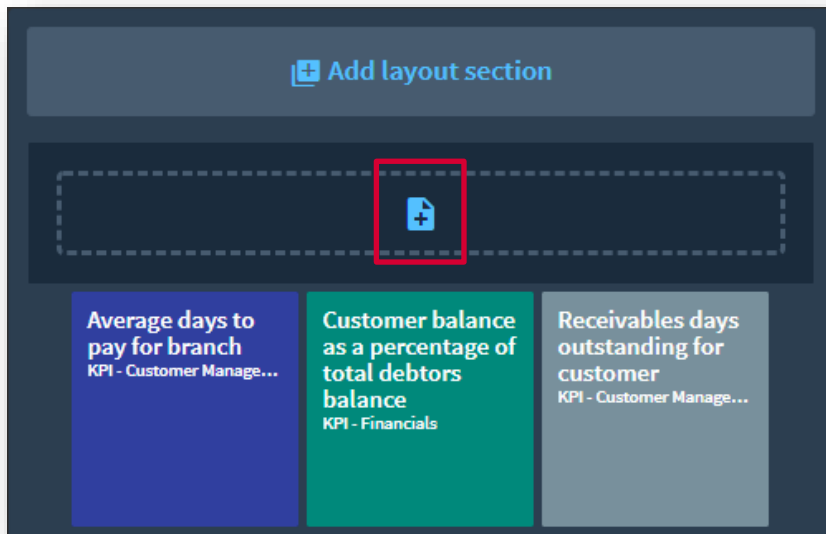
Then select Design Web View > Primary Layout:



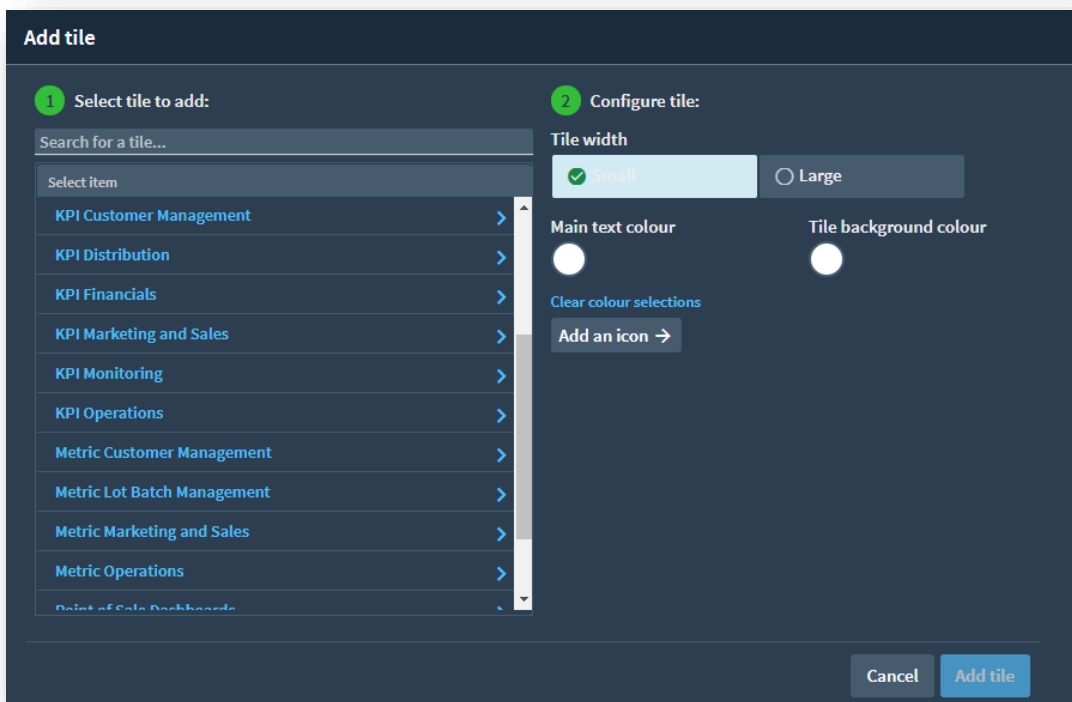
After a few seconds, the workspace editor will be shown. This is a visualization showing how the user will experience this workspace:



To add a new tile, click on the '+' symbol in the layout section.

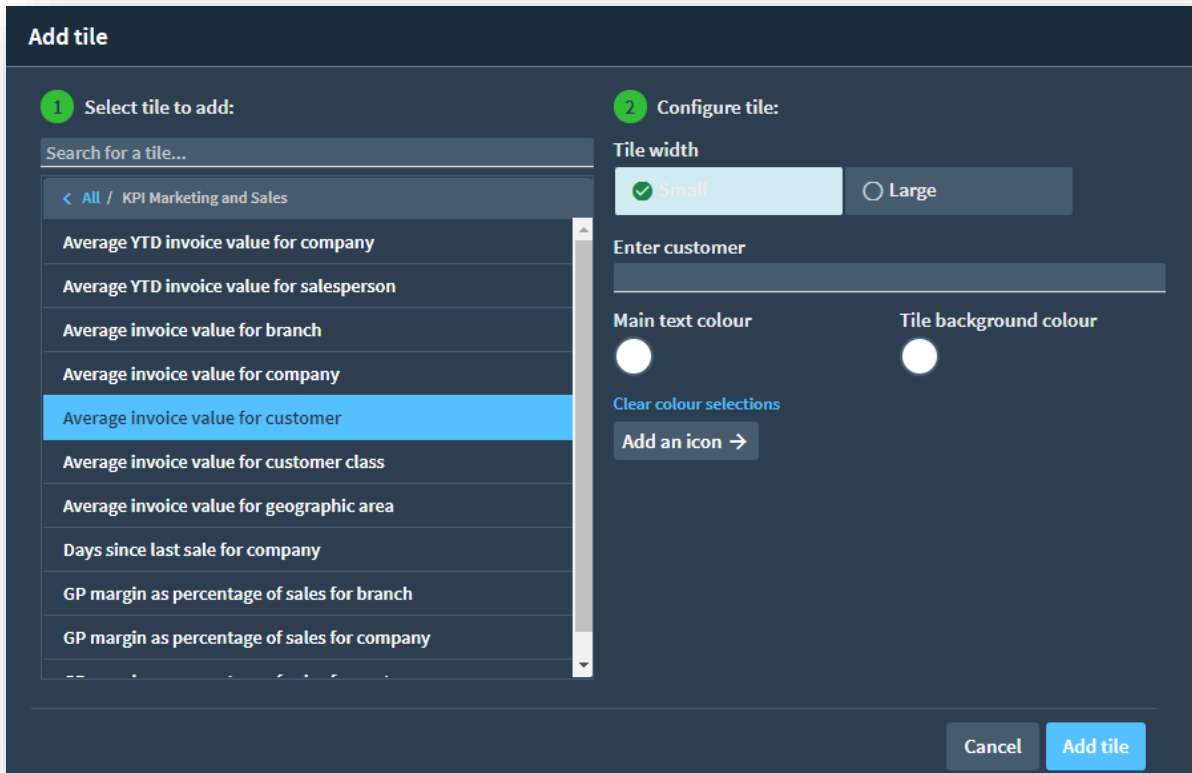


You can then locate the **Insight Tile** that you wish to add:



In this example, we are adding a tile to the Custom Query, so we should select a tile that accepts the Customer code as an input.

We are going to select a custom **Insight Tile**. We have selected the category **KPI Marketing and Sales** and the tile 'Average invoice value for customer'.



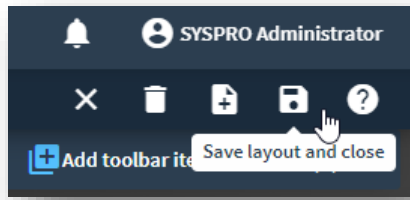
You can configure the tile by selecting the tile size (Small or Large), the tile Main text color and Tile background color and add an icon if required.

Leave the 'Enter customer' value spaces – this ensures that the customer key in context at run time will be used, rather than a hard-coded customer key.

In my case I already have several **Insight Tiles** on the workspace, so I left the new tile as the last one in the group.

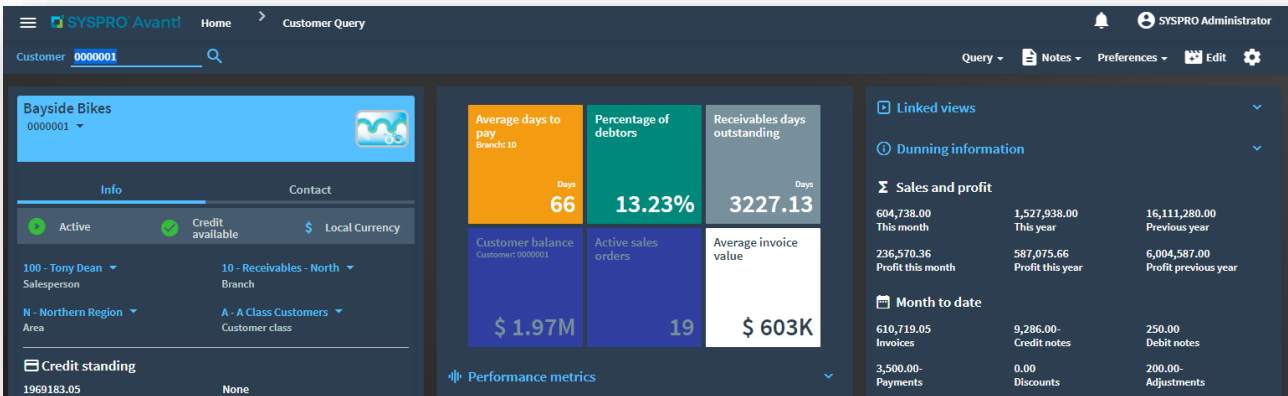


We can now save the layout and close the customization – click on the Save icon (top right of the menu bar).



Note: The application interface will not immediately be changed. You should return to the menu and then reload the Customer Query.

When you select a Customer the new **Insight Tile** will be refreshed showing the customer's average invoice value.



You can click on the **Insight Tile** to drilldown into the invoices that make up the customer's current balance.

A screenshot of a drilldown window titled 'Average invoice value'. It displays a table with the following columns: Branch, Description, Customer, Name, Invoice, Currency, Foreign value, and Exchange rate. The table contains 41 rows of data, all for 'Receivables - North' from customer '0000001' at 'Bayside Bikes'. The 'Invoice' column lists various invoice numbers, and the 'Foreign value' column shows the corresponding values in dollars. The bottom of the window indicates 'Showing 41 item(s)'.

Branch	Description	Customer	Name	Invoice	Currency	Foreign value	Exchange rate
10	Receivables - North	0000001	Bayside Bikes	100288	\$	519,200.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100293	\$	506,800.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100299	\$	121,520.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100301	\$	2,480.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100303	\$	243,280.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100305	\$	12,400.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100307	\$	560.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100312	\$	530,400.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100325	\$	44,800.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100326	\$	462,080.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100330	\$	886,868.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100337	\$	582,068.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100345	\$	1,165,600.00	1.0
10	Receivables - North	0000001	Bayside Bikes	100361	\$	1,799,200.00	1.0

Creating a Custom Tile with Context

In this topic we're going to create a new custom **Insight Tile** that shows the customer balance for the selected customer and when you click on the tile it will show a list of outstanding invoices that make up the current balance.

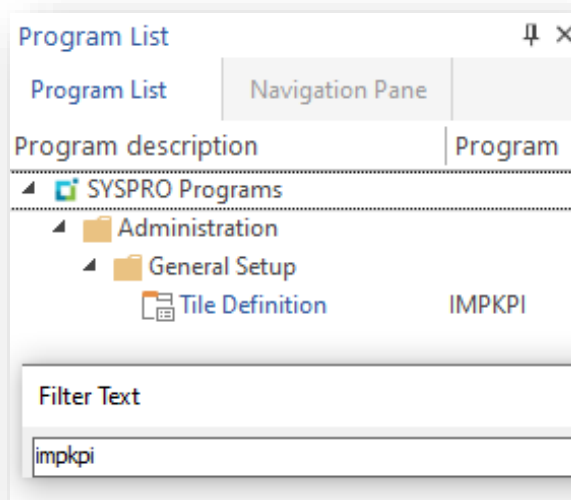
Note: When this document was first authored (early 2022), there was not yet an Avanti interface for the Insight Tile Builder application. Therefore, we will be using the SYSPRO 8 Rich Client interface for all examples.

Remember screen images may differ from that shown based on various factors.

INSIGHT TILE DEFINITION

Login to SYSPRO and from the main menu run the **Tile Definition** (IMPKPI) program

In the example below I filtered on the program list and entered 'IMPKPI':



As you will be running this program multiple times it is recommended that you add it to your Favorites menu.

When you run the application, it will show a list of all tiles, both standard tiles provided as part of the SYSPRO application, and custom tiles that you have created yourself. You can select the 'Filter Customer Tiles' toolbar checkbox to simply show the tiles that you have created.

Insight Tile Definition

File View

Refresh View Filter KPI Where Used Filter Custom Tiles New Tile Import Export

Tiles

Tile	Where used	System KPI	Company KPI	Role KPI	Operator KPI	Custom	Development
Tile category: Financial KPI							
Outstanding balance for portal supplier				Add	Add	Standard	<input type="checkbox"/>
Outstanding balance for portal supplier(copy)				Add	Add	Custom	<input type="checkbox"/>
Tile category: KPI - Customer Management							
Average days to pay for customer	View (1)			Add	Add	Standard	<input type="checkbox"/>
Receivables days outstanding for customer	View (3)			Add	Edit KPI (1)	Standard	<input type="checkbox"/>
Average days to pay for branch	View (3)			Add	Edit KPI (2)	Standard	<input type="checkbox"/>
Receivables days outstanding for branch				Add	Add	Standard	<input type="checkbox"/>
Outstanding balance for customer				Add	Add	Standard	<input type="checkbox"/>
Average days to pay for portal customer				Add	Add	Standard	<input type="checkbox"/>
Receivables days outstanding for portal customer				Add	Add	Standard	<input type="checkbox"/>
Tile category: KPI - Distribution							
Number of active quotation requests				Add	Add	Standard	<input type="checkbox"/>
Number of active submissions				Add	Add	Standard	<input type="checkbox"/>
Number quotes for the year				Add	Add	Standard	<input type="checkbox"/>
Number of quotes that are accepted to a purchase order				Add	Add	Standard	<input type="checkbox"/>
Number of quotes rejected against that supplier				Add	Add	Standard	<input type="checkbox"/>
Outstanding value for active quotes				Add	Add	Standard	<input type="checkbox"/>
Number of started missions for the day by cycle count				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding missions for the day by cycle count				Add	Add	Standard	<input type="checkbox"/>
Number of Missions Created From Cycle Count				Add	Add	Standard	<input type="checkbox"/>
Number of Missions by Pick				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding Missions created today				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding Missions created this week				Add	Add	Standard	<input type="checkbox"/>
Number of Missions completed today				Add	Add	Standard	<input type="checkbox"/>
Number of Missions completed this week				Add	Add	Standard	<input type="checkbox"/>
Tiles: 113							

A good practice is to add the **Tileid** column to this listview using the Field Chooser. See the sample below.

Insight Tile Definition

File View

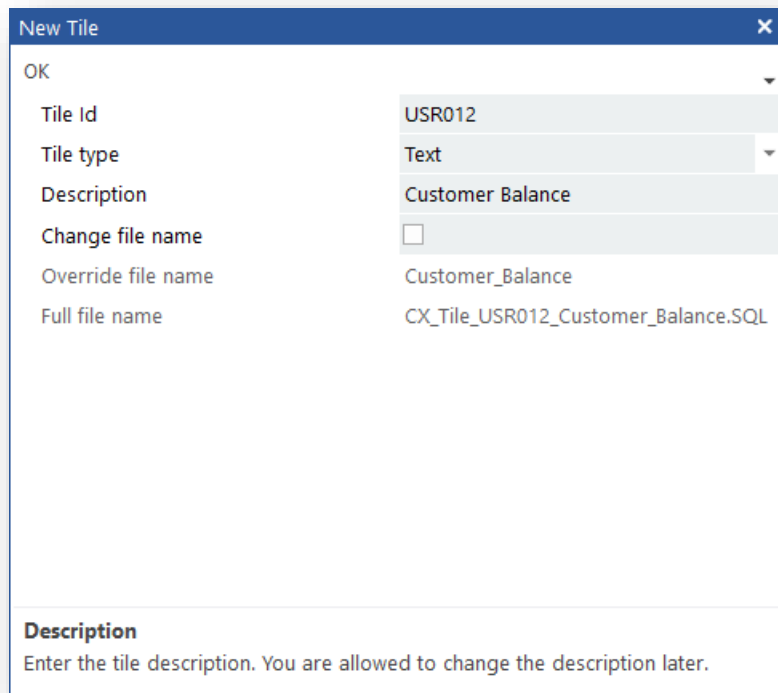
Refresh View Filter KPI Where Used Filter Custom Tiles New Tile Import Export

Tiles

Tile	Tileid	Where used	System KPI	Company KPI	Role KPI	Operator KPI	Custom	Development
Tile category: Financial KPI								
Outstanding balance for portal supplier	APS001				Add	Add	Standard	<input type="checkbox"/>
Outstanding balance for portal supplier(copy)	USR003				Add	Add	Custom	<input type="checkbox"/>
Tile category: KPI - Customer Management								
Average days to pay for customer	ARS014	View (1)			Add	Add	Standard	<input type="checkbox"/>
Receivables days outstanding for customer	ARS016	View (3)			Add	Edit KPI (1)	Standard	<input type="checkbox"/>
Average days to pay for branch	ARS022	View (3)			Add	Edit KPI (2)	Standard	<input type="checkbox"/>
Receivables days outstanding for branch	ARS026				Add	Add	Standard	<input type="checkbox"/>
Outstanding balance for customer	ARS028				Add	Add	Standard	<input type="checkbox"/>
Average days to pay for portal customer	ARS029				Add	Add	Standard	<input type="checkbox"/>
Receivables days outstanding for portal customer	ARS030				Add	Add	Standard	<input type="checkbox"/>
Tile category: KPI - Distribution								
Number of active quotation requests	RFQ001				Add	Add	Standard	<input type="checkbox"/>
Number of active submissions	RFQ002				Add	Add	Standard	<input type="checkbox"/>
Number quotes for the year	RFQ003				Add	Add	Standard	<input type="checkbox"/>
Number of quotes that are accepted to a purchase order	RFQ004				Add	Add	Standard	<input type="checkbox"/>
Number of quotes rejected against that supplier	RFQ005				Add	Add	Standard	<input type="checkbox"/>
Outstanding value for active quotes	RFQ006				Add	Add	Standard	<input type="checkbox"/>
Number of started missions for the day by cycle count	WHM001				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding missions for the day by cycle count	WHM002				Add	Add	Standard	<input type="checkbox"/>
Number of Missions Created From Cycle Count	WHM003				Add	Add	Standard	<input type="checkbox"/>
Number of Missions by Pick	WHM004				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding Missions created today	WHM005				Add	Add	Standard	<input type="checkbox"/>
Number of outstanding Missions created this week	WHM006				Add	Add	Standard	<input type="checkbox"/>
Number of Missions completed today	WHM007				Add	Add	Standard	<input type="checkbox"/>
Number of Missions completed this week	WHM008				Add	Add	Standard	<input type="checkbox"/>
Tiles: 114								

NEW TILE DIALOG

Click on 'New Tile' toolbar button to load the New Tile dialog.



OK	
Tile Id	USR012
Tile type	Text
Description	Customer Balance
Change file name	<input type="checkbox"/>
Override file name	Customer_Balance
Full file name	CX_Tile_USR012_Customer_Balance.SQL

Description
Enter the tile description. You are allowed to change the description later.

A unique Tile id will be generated and shown – you can change this if required. In this example we will retain the generated tile id of 'USR012'.

Select the 'Text' Tile type and enter a tile description so that you can find this tile later. The tile description is not shown on the tile but will be used as the default tile title.

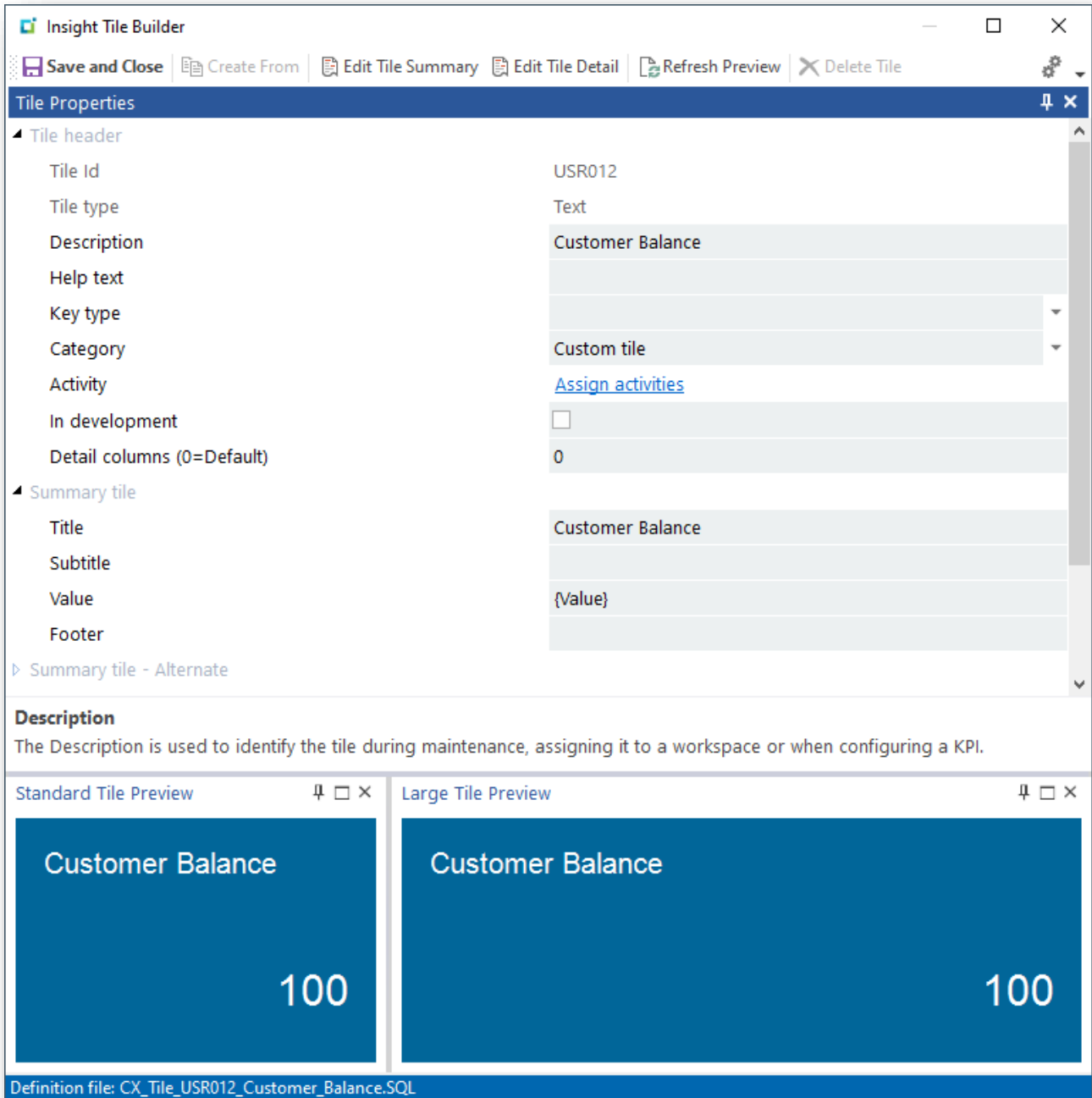
You can leave the other options as their defaults.

Click 'OK' or press Enter to start defining the tile properties.

Warning: The tile id, tile type and full file name cannot be changed later.

INSIGHT TILE BUILDER - TILE PROPERTIES

The Insight Tile Builder - Tile Properties dialog will be shown with defaults applied:



Note that the Category defaults to 'Custom tile' and the default Title is set to the tile description. We will retain this Category as it makes it easier to later select custom tiles – but you can change your category as required.

The help text is used to provide a more detailed description of the tile. This can be useful when reviewing and maintaining the custom tile in future.

As we are defining a context tile (where a key field will be passed to the tile for selection purposes), we should select one of the standard SYSPRO keys using the 'Key type' dropdown. In our example we will use **Customer**.

Lastly you can set the tile 'In development' flag to checked. This prevents anyone adding this tile to a user's workspace. We will unset this once we have completed this tile definition.

The Tile Properties now look as follows – the relevant fields have been highlighted:

The screenshot shows the 'Insight Tile Builder' application window. The 'Tile Properties' section is expanded, showing the following fields and values:

Property	Value
Tile Id	USR012
Tile type	Text
Description	Customer Balance
Help text	Show the customer current balance with a drilldown to the cur
Key type	Customer
Category	Custom tile
Activity	Assign activities
In development	<input checked="" type="checkbox"/>
Detail columns (0=Default)	0
Summary tile	
Title	Customer Balance
Subtitle	
Value	{Value}
Footer	

CATEGORY

When tiles are shown in the **Insight Tile Definition** program, they are grouped by Category. Similarly, when adding an **Insight Tile** to a user's workspace, they are grouped by Category. In both cases this makes it much easier to find and select tiles as you do not have to scroll through a potentially very long list of tiles.

The Category is a free-format field allowing you to use any relevant description.

To help you categorize tiles consistently, the dropdown shows all currently defined categories. This list includes custom categories that you have defined, and standard ones shipped as part of the SYSPRO application.

It is recommended that you prefix any user defined categories with the word 'Custom'. This just makes it easier to differentiate between standard and custom tiles.

ACTIVITY

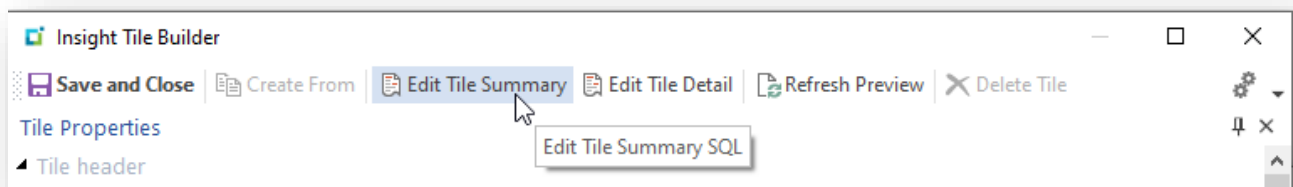
You can optionally assign one or more business activities to the tile. This subject is not covered in this example.

For more information see the topic: [Insight Tile Properties – Business Activities](#)

SUMMARY SQL

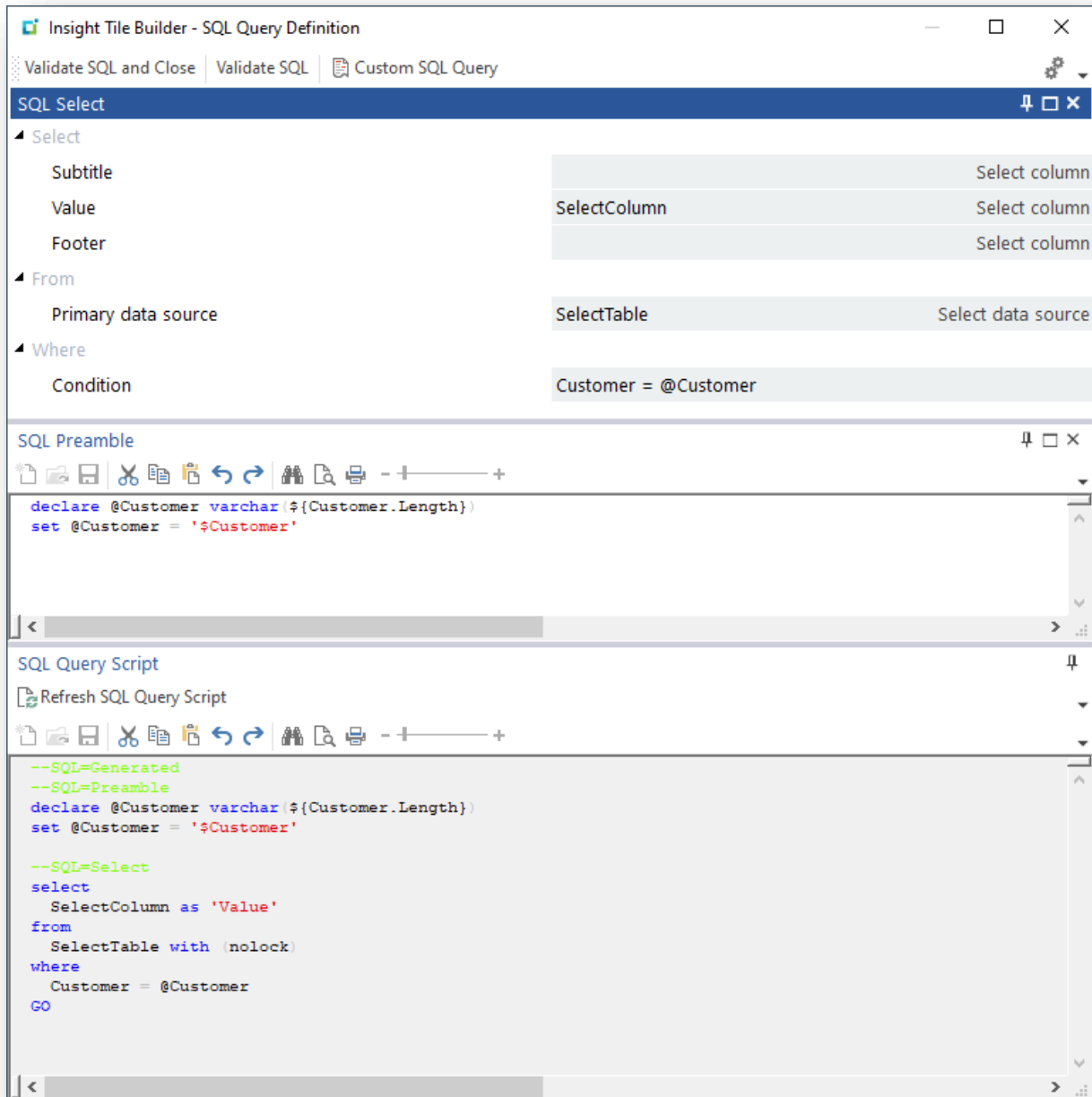
Now we are ready to define the Summary SQL statement that will be used to retrieve the customer's current balance.

Select the Edit Tile Summary toolbar button.



This loads the **Insight Tile Builder – SQL Query Definition** dialog which allows you to define the Tile Summary SQL statement that will be used at run time.

The default will be shown as follows:



The purpose of this dialog is to define a SQL statement that returns a single row with 1, 2 or 3 columns of data. Each column can be shown on the text tile. You must return a mandatory column named 'Value'.

Note: The **as 'Value'** clause in the generated SQL script means that regardless of the column definition used to select the data, the column name returned to the application is the word 'Value'.

Optionally you can also return columns named 'SubTitle' and/or 'Footer'.

Important: All column names are case sensitive.

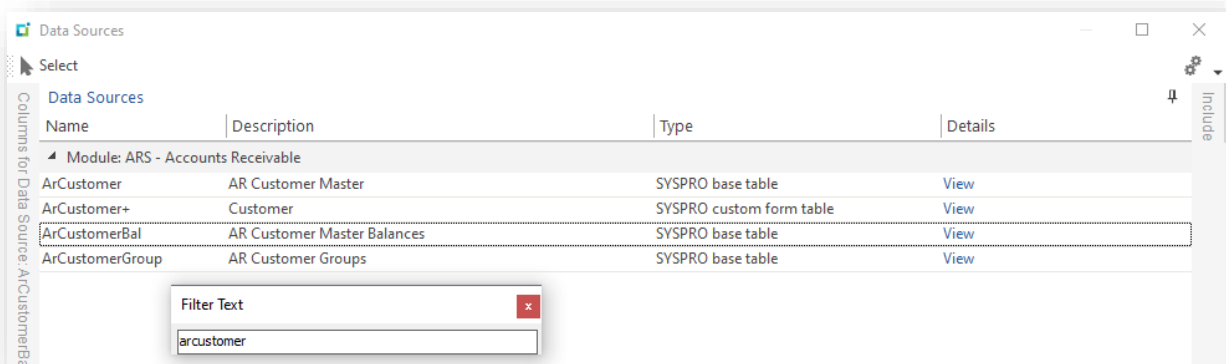
SUMMARY GENERATED SQL

You can use the dialog to generate a summary SQL statement without any expert SQL knowledge.

If you wish to create a more sophisticated SQL statement, you can use the Custom SQL definition dialog. This is explained later – see the topic: [Summary Custom SQL](#)

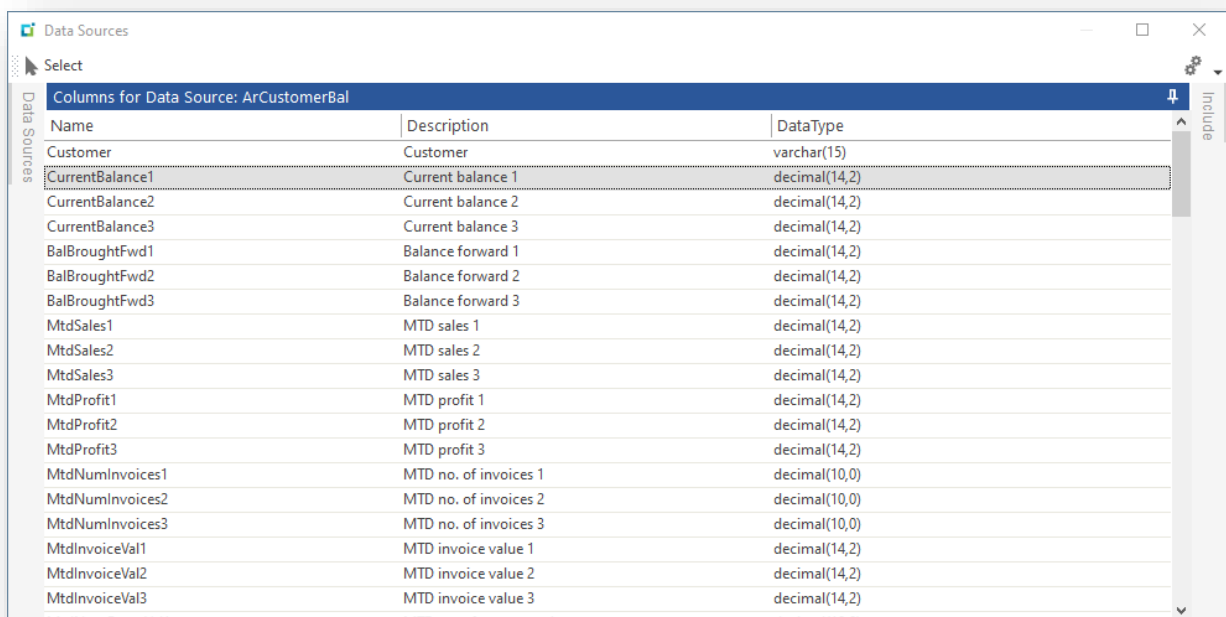
The simplest method of defining the SQL statement is first to select the **Primary data source**. Use the 'Select data source' button and select a SQL base table, a custom form table, a SYSPRO business view or a user defined table or view.

In our example we're going to select the **ArCustomerBal** SYSPRO base table.

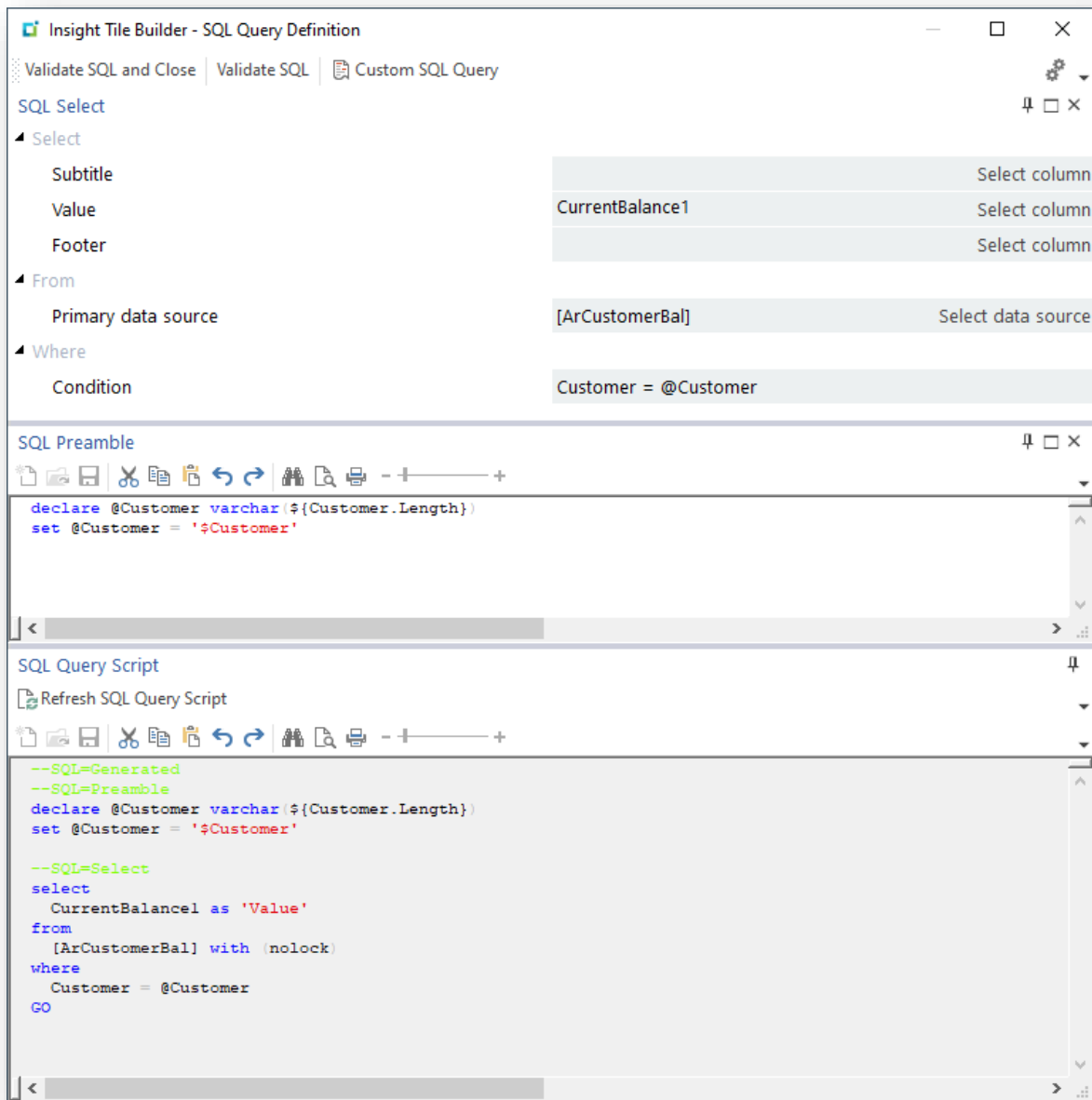


Once you have selected the data source you should select the column against the **Value** prompt by clicking on the 'Select column' button.

The Data source selection program will use the Primary data source and only show relevant columns. In our example we're going to select the column named **CurrentBalance1**.



Once selected, the SQL Query Definition dialog will look as follows:



Note: We have selected column **CurrentBalance1** from the data source **ArCustomerBal**. This is a standard SYSPRO base table defined in the current company to which the user is connected. The data source is shown in brackets.

As we previously selected the 'Customer' Key type when creating the tile, the condition was pre-populated with:

```
Customer = @Customer
```

This indicates that the SQL 'where' clause will compare the column named **Customer** with a SQL variable named **@Customer**.

A **SQL Preamble** has been pre-populated with the SQL statements:

```
declare @Customer varchar({Customer.Length})
set @Customer = '$Customer'
```

The 'declare' statement defines a SQL variable named **@Customer** with datatype **varchar(15)** – the variable **#{Customer.Length}** is expanded at run time to the length of the customer key which is 15.

The 'set' statement, sets the SQL variable **@Customer** to the value of the current customer passed in context from the user interface. i.e. the variable **#{Customer}** is replaced at run time with the current customer code from the application on which the tile has been configured.

For example, if you have numeric customer codes and the current customer is '0000001' then the **SQL Preamble** will be expanded to:

```
declare @Customer varchar(15)
set @Customer = '000000000000001'
```

You can then use the SQL variable **@Customer** in the SQL statement.

In the 'Generated SQL' mode, the final SQL statement being generated is shown at the bottom of the properties dialog.

```
--SQL=Generated
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '$Customer'

--SQL=Select
select
  CurrentBalance1 as 'Value'
from
  [ArCustomerBal] with (nolock)
where
  Customer = @Customer
GO
```

Note: There are some special formatted comments that begin '--SQL='. The line '--SQL=Generated' indicates to the Insight Tile Builder application that this SQL statement was generated using this dialog.

The table hint **with (nolock)** is automatically appended to the table name to ensure that this query statement is not blocked by any database transactions. This is important to ensure that the user interface does not block (appear to hang) if a transaction is in progress affecting the selected data.

Once you have defined the SQL statement, click the 'Verify SQL and Close' toolbar button. This performs some basic validation on the generated SQL statement. If an error is returned, make any corrections, and retry. When successful you should receive the following notification.

SQL Validation Message

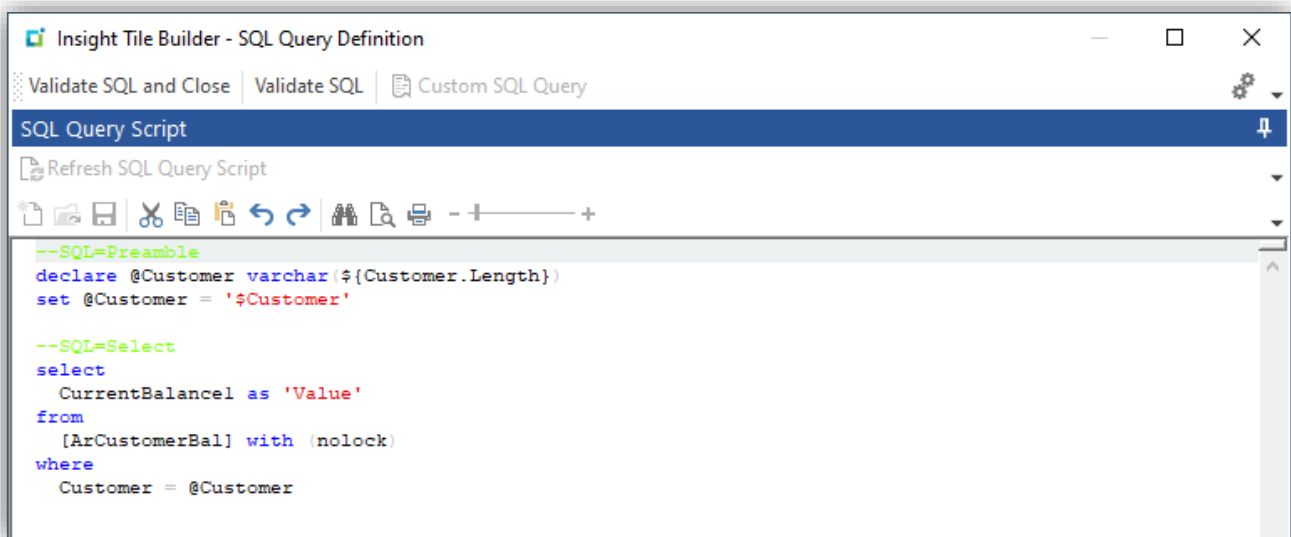
SQL validation successful.

Once you have defined the Tile Summary SQL, we are ready to define the Tile Detail SQL. See the topic [Tile Detail SQL](#).

SUMMARY CUSTOM SQL

If required, you can customize the Summary SQL statement that has been generated.

Click on the 'Custom SQL Query' toolbar button, the SQL Query Script syntax editor will be enabled allowing you to define virtually any appropriate SQL statement.



```
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '{Customer}'

--SQL=Select
select
    CurrentBalance1 as 'Value'
from
    [ArCustomerBal] with (nolock)
where
    Customer = @Customer
```

The special comments indicating the SQL Preamble and SQL Select statement are retained – these can be removed if required and do not form part of the custom SQL query.

When defining a custom SQL query statement, you should be aware of the following:

- You must return a single row with a column named 'Value' (exact case)
- You can optionally return columns named 'SubTitle' and 'Footer' (exact case)
 - It is not recommended that you return additional columns
 - The only exception is an optional column named 'KpiValue'
 - See [KPI Threshold Comparison: Value and KpiValue](#)
- You can use variables that are passed from the SYSPRO **Insight Tile** interface and will be replaced at run time with the relevant value. These consist of:
 - \${Variable} – system variables such as \${Operator}, \${SystemDb} or \${CompanyDb}

- See [Appendix 3 - System environment variables](#)
- `${Key.Length}` – key length variables such as `${Customer.Length}`
 - See [Appendix 1 - List of Key Types](#)
- `$Variable` – the value of these variables depends on the application context for example `$Customer` expands to the customer key – including leading zeros if numeric.
 - See [Appendix 1 - List of Key Types](#)
- The default database context will be the current company database.
- You must use the **with (noLOCK)** table hint against all source tables. This ensures that the user interface does not appear to hang (wait) if a transaction is in progress.
- Source tables can include:
 - SYSPRO base tables
 - SYSPRO custom form tables
 - SYSPRO business views
 - User defined tables
 - User defined views

Once you have defined the SQL statement, click 'Verify SQL and Close' toolbar button. This performs some basic validation on the custom SQL statement. If an error is returned, make any corrections, and retry. When successful you should receive a SQL Validation Message.

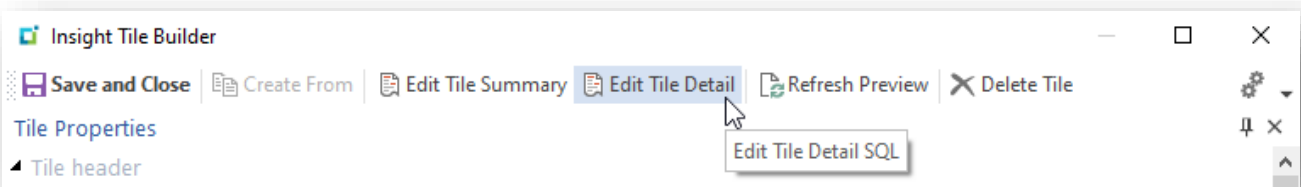
Once you have defined the Tile Summary SQL, we are ready to define the Tile Detail SQL. See below.

DETAIL SQL

Now we are ready to define the Detail SQL statement that will be used to retrieve a list of all the invoices that make up the current customer balance.

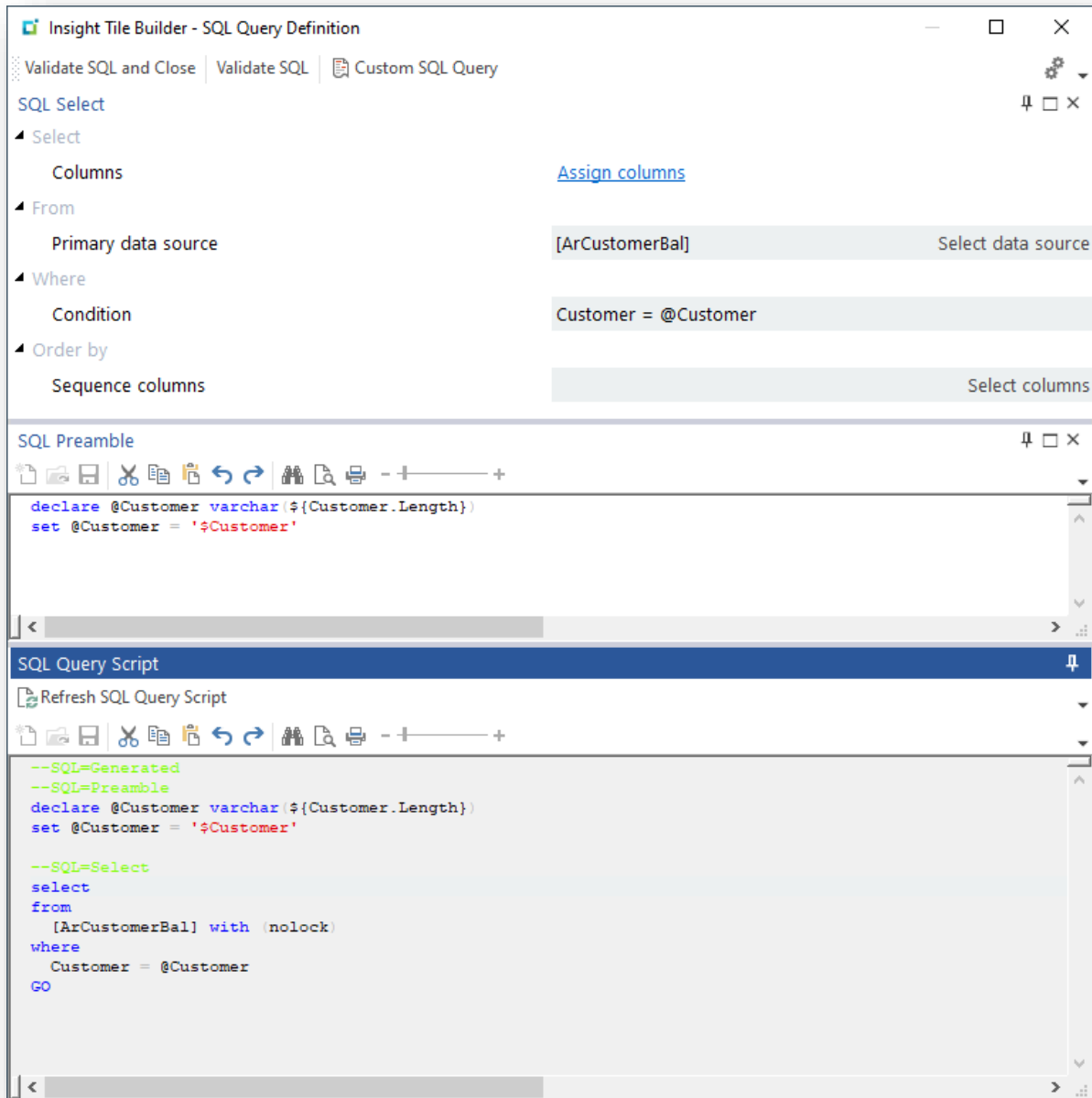
A list is used to present the detail drilldown results – the Detail SQL statement must return the appropriate set of rows and for each row a set of columns to be shown.

Select the Edit Tile Detail toolbar button.



This will load the **Insight Tile Builder – SQL Query Definition** dialog allowing you to define the Tile Detail SQL statement that will be used at run time.

The default will be shown as follows:



The purpose of this dialog is to define a SQL statement that returns rows that will be shown as the drilldown detail when clicking on the summary tile.

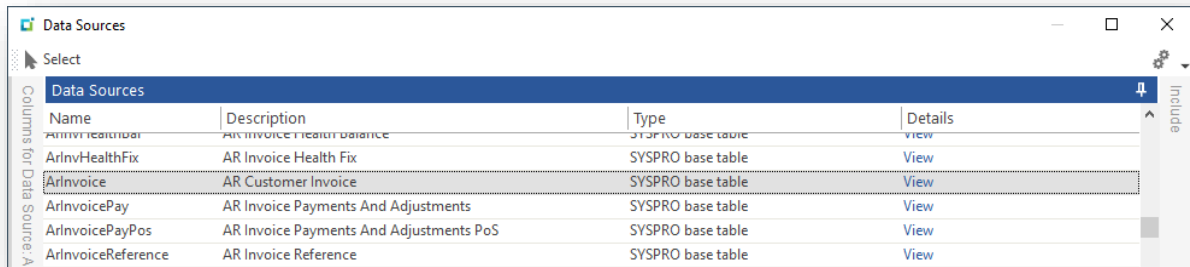
You can use the dialog to generate a detail SQL statement without any expert SQL knowledge.

If you wish to create a more sophisticated SQL statement, you can use the Custom SQL definition dialog. This is explained later – see the topic: [Detail Custom SQL](#)

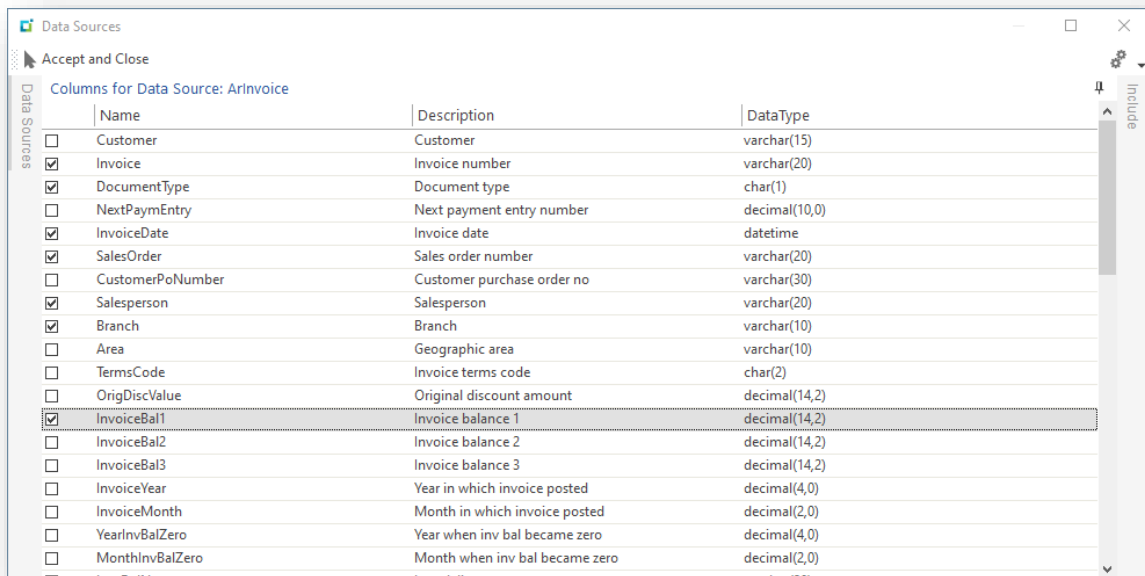
DETAIL GENERATED SQL

The simplest method of defining the SQL statement is first to select the **Primary data source**. Use the 'Select data source' button and select a SQL base table, a custom form table, a SYSPRO business view or a user defined table or view.

This defaults to the same as the Summary SQL data source. However, in this example we want to return rows from the customer Invoice table. Therefore, first use the 'Select data source' button and select the table **ArInvoice**.



Then click on the 'Assign columns' hyperlink against the Column prompt. You will be presented with the data sources selection dialog allowing you to select one or more columns to be returned.



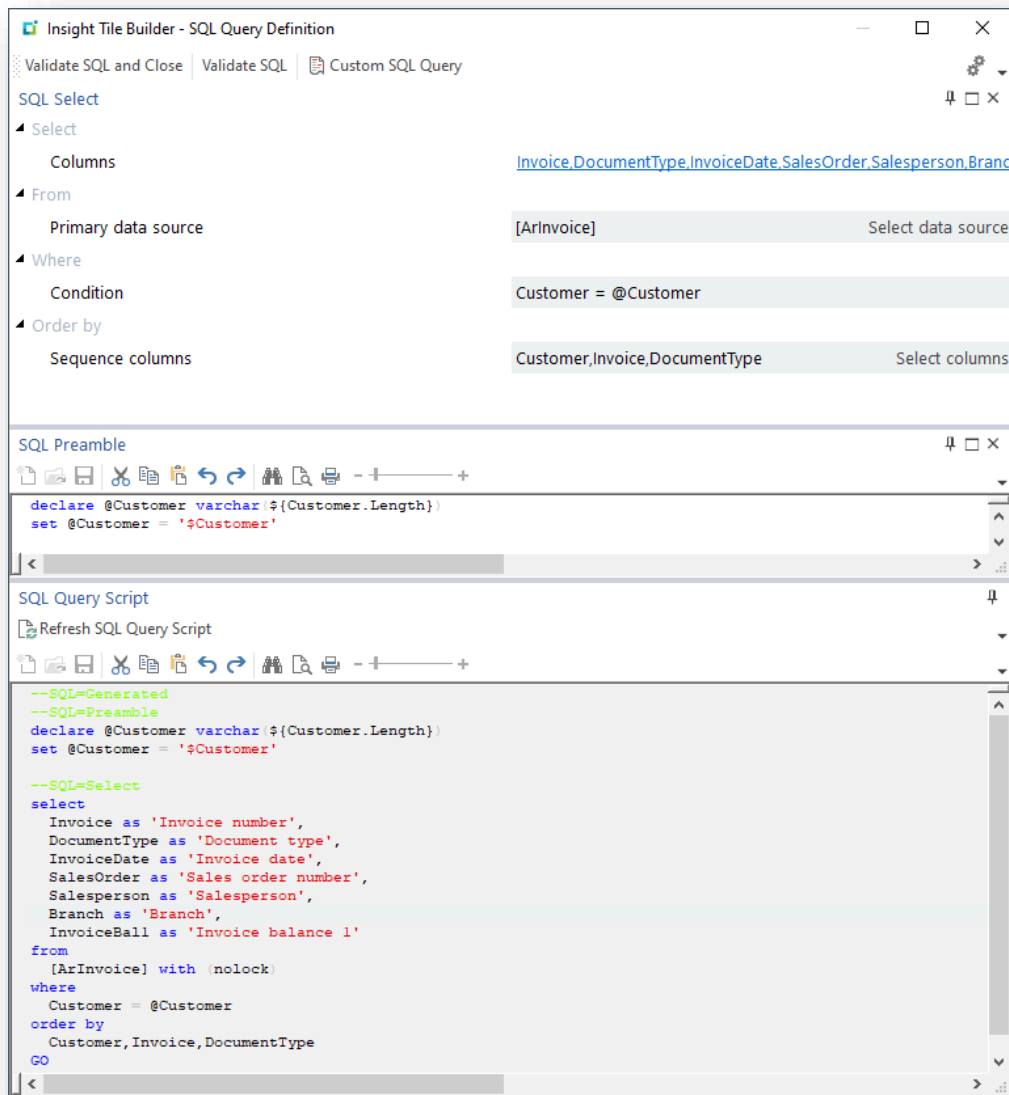
In this example we have selected the following columns:

- Invoice
- DocumentType
- InvoiceDate
- SalesOrder
- Salesperson
- Branch
- InvoiceBal1

You can make any relevant selection.

See the topic: [Detail SQL – Keeping the initial dropdown list simple](#)

Once selected the SQL Query Definition dialog will show the selected columns as a comma separated hyperlink – as shown below:



Note: We have selected several columns from the data source **ArInvoice**. This is a standard SYSPRO table defined in the current company to which the user is connected. The data source is shown in brackets. i.e. **[ArInvoice]**

The condition was pre-populated with:

```
Customer = @Customer
```

This indicates that the SQL 'where' clause will compare the column named **Customer** with a SQL variable named **@Customer**.

By default, the rows returned will not be sorted into any specific sequence. You can override this behavior by creating a comma separated list of columns that will be used to sequence the output of the select statement – this becomes the **order by** clause. In this example we have defined:

```
Customer, Invoice, DocumentType
```

A **SQL Preamble** has been pre-populated with the SQL statements:

```
declare @Customer varchar(${Customer.Length})
set @Customer = '$Customer'
```

This is identical to the summary SQL logic and will not be repeated here.

In the 'Generated SQL' mode, the final SQL statement being generated is shown at the bottom of the properties dialog.

```
--SQL=Generated
--SQL=Preamble
declare @Customer varchar(${Customer.Length})
set @Customer = '$Customer'

--SQL=Select
select
  Invoice as 'Invoice number',
  DocumentType as 'Document type',
  InvoiceDate as 'Invoice date',
  SalesOrder as 'Sales order number',
  Salesperson as 'Salesperson',
  Branch as 'Branch',
  InvoiceBall as 'Invoice balance 1'
from
  [ArInvoice] with (nolock)
where
  Customer = @Customer
order by
  Customer, Invoice, DocumentType
GO
```

Note: There are some special formatted comments that begin '--SQL='. The line '--SQL=Generated' indicates to the Insight Tile Builder application that this SQL statement was generated using this dialog.

The table hint **with (nolock)** is automatically appended to the table name to ensure that this query statement is not blocked by any database transactions. This is important to ensure that the user interface does not block (appear to hang) if a transaction is in progress affecting the selected data.

Once you have defined the SQL statement, click 'Verify SQL and Close' toolbar button. This performs some basic validation on the generated SQL statement. If an error is returned, make any corrections, and retry. When successful you should receive a SQL Validation Message.

DETAIL SQL – KEEPING THE INITIAL DRILLDOWN LIST SIMPLE

It is good practice to configure the detail drilldown so that only the most important columns are returned by default. This means that the detail drilldown looks simple when first used.

However, in some examples it could be useful to return a larger number of columns and allow the end-user to select the ones they wish to see.

In this case, you should use the Detail SQL statement to return all the columns you believe could be useful, and then use the Insight Tile Properties option: **Detail columns (0=Default)** prompt to configure the initial number of columns that are shown to the user.

For example, suppose you configure that there are (say) 12 columns that you believe could be useful to the end user but to keep the initial detail drilldown simple you only show 5 columns.

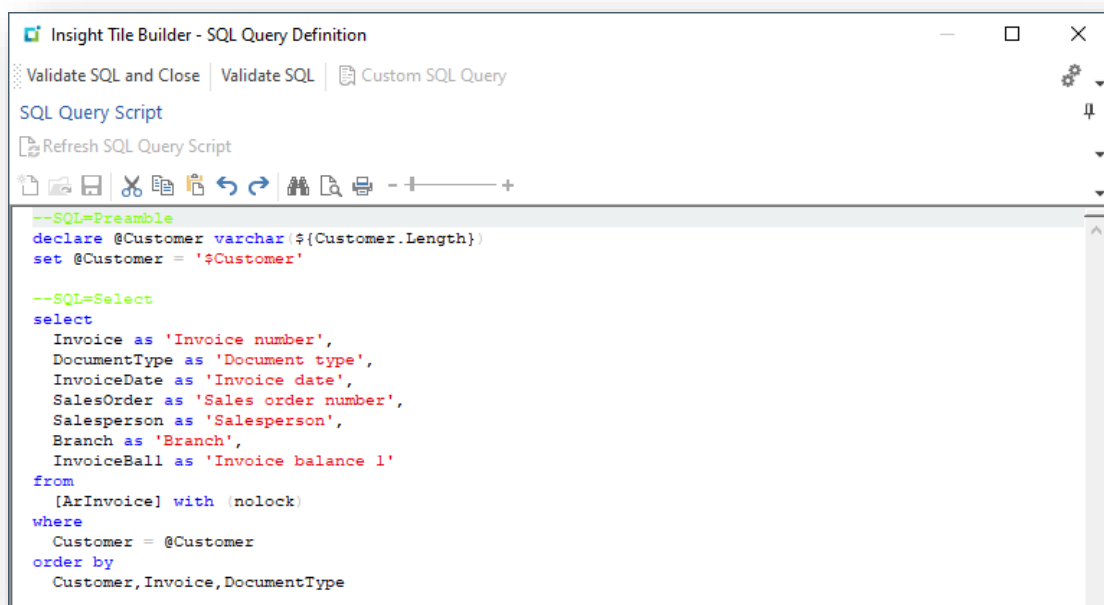
Then set the **Detail columns (0=Default)** prompt to 5.

At run time the user will be presented with the initial 5 columns or they can use the Field Chooser to select from any of the 12 available columns.

DETAIL CUSTOM SQL

If required, you can customize the Detail SQL statement that has been generated.

Click on the 'Custom SQL Query' toolbar button, the SQL Query Script syntax editor will be enabled allowing you to define virtually any relevant SQL statement.



```
Insight Tile Builder - SQL Query Definition
Validate SQL and Close | Validate SQL | Custom SQL Query
SQL Query Script
Refresh SQL Query Script
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '{Customer}'
--SQL=Select
select
    Invoice as 'Invoice number',
    DocumentType as 'Document type',
    InvoiceDate as 'Invoice date',
    SalesOrder as 'Sales order number',
    Salesperson as 'Salesperson',
    Branch as 'Branch',
    InvoiceBall as 'Invoice balance 1'
from
    [ArInvoice] with (nolock)
where
    Customer = @Customer
order by
    Customer, Invoice, DocumentType
```

The special comments indicating the SQL Preamble and SQL Select statement are retained – these can be removed if required and do not form part of the custom SQL query.

When defining a custom SQL query statement, you should be aware of the following:

- You must return at least one column.
 - Remember to keep the number of columns to a low number or use the Insight Tile Properties 'Detail Columns' option to limit the columns.
- The condition (where statement) should use selection logic to ensure that the rows returned are providing the detail representing the value shown on the summary tile.
- You can use variables that are passed from the SYSPRO **Insight Tile** interface and will be replaced at run time with the relevant value. These consist of:
 - `${Variable}` – system variables such as `${Operator}`, `${SystemDb}` or `${CompanyDb}`
 - See [Appendix 3 - System environment variables](#)
 - `${Key.Length}` – key length variables such as `${Customer.Length}`
 - See [Appendix 1 - List of Key Types](#)
 - `$Variable` – the value of these variables depends on the application context for example `$Customer` expands to the customer key – including leading zeros if numeric.
 - See [Appendix 1 - List of Key Types](#)
- The default database context will be the current company database.
- You must use the **with (nolock)** table hint against all source tables. This ensures that the user interface does not appear to hang (wait) if a transaction is in progress.
- Source tables can include:
 - SYSPRO base tables
 - SYSPRO custom form tables
 - SYSPRO business views
 - User defined tables
 - User defined views
- The 'as' clause allows you to define the detail list column captions.
 - Each 'as' clause has a maximum of 30 characters.
 - There are some modifiers that can be appended to the description to modify how the column is rendered – these include:
 - `{Guid}`
 - `{String}`
 - `{Number.d}`
 - `{Time}`
 - `{Time.f}`
 - `{Datm}`

See the topic: [Detail SQL – Keeping the initial dropdown list simple](#)

Once you have defined the SQL statement, click 'Verify SQL and Close' toolbar button. This performs some basic validation on the custom SQL statement. If an error is returned, make any corrections, and retry. When successful you should receive a SQL Validation Message.

TEXT TILE PREVIEW

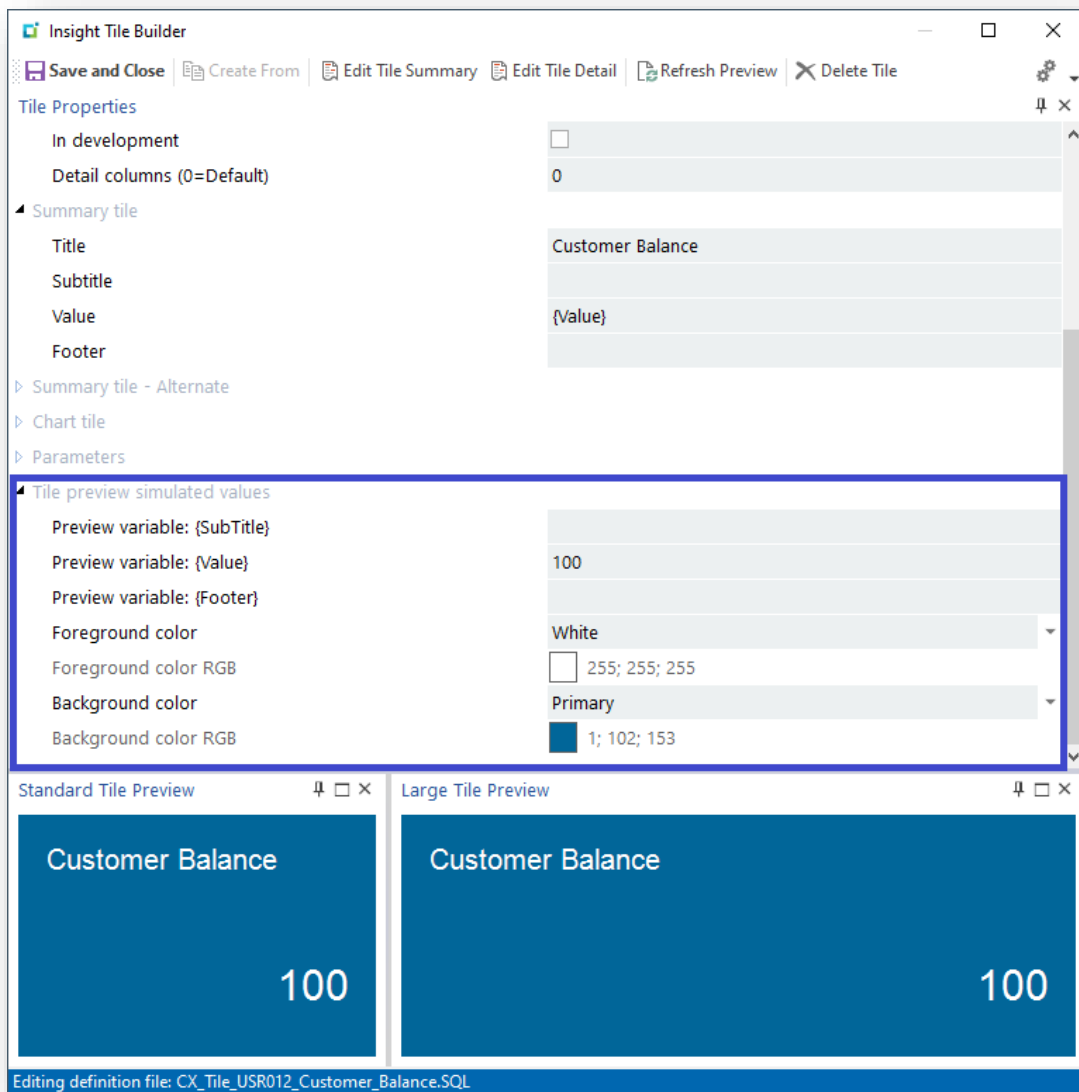
When creating and maintaining a text tile you can use the tile preview to help visualize how the tile will look at run time.

Two previews are provided, one for the small tile and the other for the large tile preview.

TILE PREVIEW - Simulation

The tile previews are not intended to exactly represent how the tile will look at run time as depending on your user interface (Web UI or Windows Rich client), any applied theme, the exact screen size, scaling and other properties it may appear differently.

If required, you can adjust how the preview looks by scrolling down the main Insight Tile Builder properties and making appropriate selections under the section named 'Tile preview simulated values' – see below:



TILE PREVIEW - DOES NOT SHOW REAL DATA

No SQL statements are issued to retrieve data when showing the tile previews. The values shown on the preview are taken from the Summary Tile > Title value and the entries for subtitle, value and footer as defined in this dialog.

This allows you to preview how any relevant string will look, including more sophisticated formatting options explained later.

There are several reasons for this – they include:

- You may not yet have defined the summary SQL statement, or it is still a work in progress
- You may be defining this tile on an environment, such as a test system, that does not have the data required to retrieve appropriate values
- The SQL statement may depend on data in context, such as a key or database name, that is only available on a live system and is not available when developing the tile
- The tile may be showing sensitive data and it is not appropriate to show this data to the user creating or maintaining the tile

The preview simulated values are stored against the tile definition. If you export the tile and later import it, the preview simulated values will also be imported.

TILE PREVIEW – TEXT TILES ONLY

The tile previews only simulates a summary text tile.

The detail view is not simulated, nor is a chart tile.

SAVE AND ADD TO A WORKSPACE

Once you have completed the tile definition and you are ready to save it and add it to a user's workspace.

DESELECT THE 'IN DEVELOPMENT' FLAG AND SAVE

Deselect the 'In development' attribute and use the 'Save and Close' toolbar button.

The screenshot displays the 'Insight Tile Builder' application window. The top toolbar includes buttons for 'Save and Close', 'Create From', 'Edit Tile Summary', 'Edit Tile Detail', 'Refresh Preview', and 'Delete Tile'. The 'Tile Properties' panel is open, showing the following configuration:

- Tile header**
 - Tile Id: USR012
 - Tile type: Text
 - Description: Customer Balance
 - Help text: Show the customer current balance with a drilldown to the cur
 - Key type: Customer
 - Category: Custom tile
 - Activity: [Assign activities](#)
 - In development: (unchecked)
 - Detail columns (0=Default): 0
- Summary tile**
 - Title: Customer Balance
 - Subtitle:
 - Value: (Value)
 - Footer:
- Summary tile - Alternate**

Below the properties panel, there is an 'In development' section with the instruction: 'Select the In development checkbox to prevent the tile from being added to a workspace.' At the bottom, two preview windows are shown: 'Standard Tile Preview' and 'Large Tile Preview'. Both previews display a blue tile with the text 'Customer Balance' and the value '100'. The status bar at the bottom indicates the editing file: 'CX_Tile_USR012_Customer_Balance.SQL'.

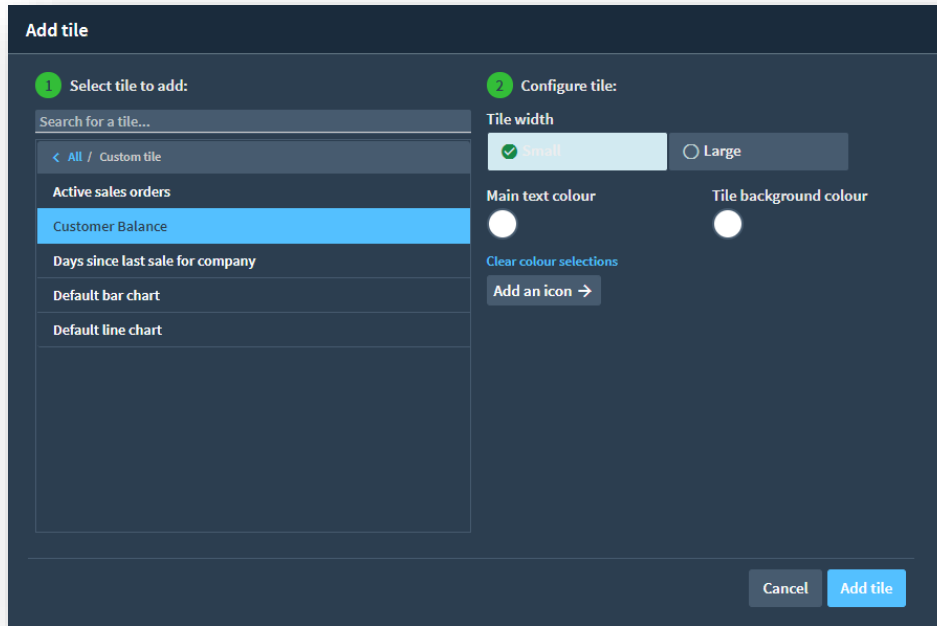
Your new tile will appear listed against the selected Category – in this example 'Custom tile'.

ADD NEW TILE TO USER'S WORKSPACE

You are now ready to add the custom tile to a user's workspace.

For example:

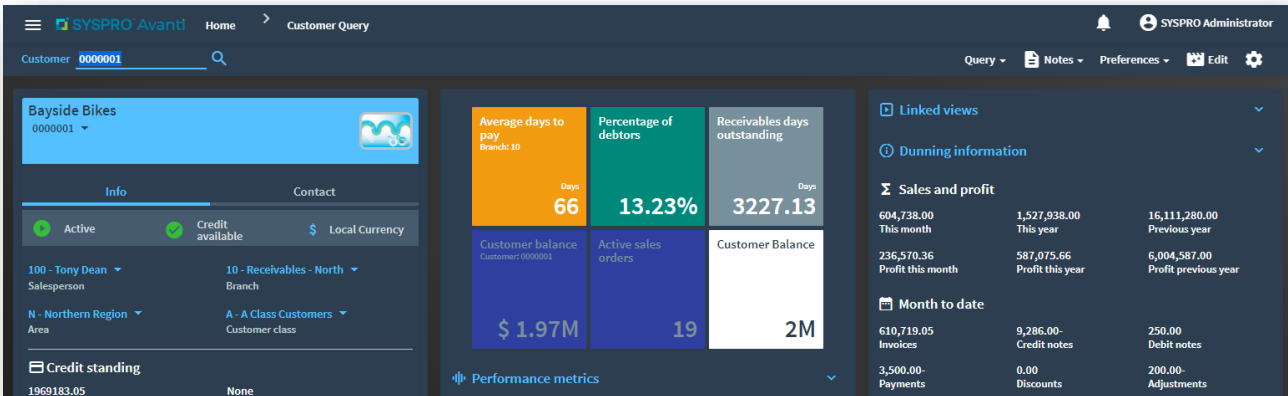
In the Customer Query, customize the user's primary workspace and select the new tile – remember it can be found under the Custom tile category:



Position the tile where required on the workspace – in this example I have added bottom right to a list of tiles already added against the Customer Query:



Save and close the layout and exit back to the menu. Then reload the Customer Query and enter a customer.



Note the Customer balance is shown using an auto-numbering style as '2M' (2 million).

You can change how this value is shown (for example using a currency symbol, change the rounding, or show the value with all digits) by editing the custom tile.

This is more fully explained in the topic: [Text Tile Formatting](#)

DRILLDOWN DETAIL


Click on the tile to show the detail drilldown:

The screenshot shows the 'Customer Balance' drilldown window, which displays a table of invoice details for the current customer. The table has the following columns: Invoice number, Document type, Invoice date, Sales order number, Salesperson, and Branch. The data is as follows:

Invoice number	Document type	Invoice date	Sales order number	Salesperson	Branch
000000000100288	I	18/03/2013	000000000000650	100	10
000000000100293	I	19/04/2013	000000000000655	100	10
000000000100299	I	22/04/2013	000000000000662	100	10
000000000100301	I	26/04/2013	000000000000662	100	10
000000000100303	I	03/05/2013	000000000000663	100	10
000000000100305	I	27/05/2013	000000000000655	100	10
000000000100307	I	27/05/2013	000000000000663	100	10
000000000100312	I	03/06/2013	000000000000668	100	10
000000000100325	I	26/07/2013	000000000000672	100	10
000000000100326	I	08/08/2013	000000000000676	100	10
000000000100330	I	09/09/2013	000000000000679	100	10
000000000100337	I	11/10/2013	000000000000685	100	10
000000000100345	I	08/11/2013	000000000000691	100	10
000000000100361	I	30/12/2013	000000000000697	100	10

The window also includes a search filter and a status bar at the bottom indicating 'Showing 41 item(s)'.

This shows the columns from the Customer Invoice table for the current customer.



You can customize the columns shown, including ensuring that the Invoice balance is shown on the initial view without having to scroll, only showing non-zero balance invoices, etc. This requires you to customize the detail SQL by changing the columns selected, SQL conditions etc.

REVIEW, CHANGE AND REVIEW AGAIN

Once you have created the **Insight Tile** and verified it works as expected, you will often return to the Insight Tile Builder to make further changes to ensure the tile is as effective as possible.

You can customize how the value is shown using various formatting techniques, the columns returned in the drilldown list and their formatting.

These customizations are covered in various sections later in this document including [Insight Tile Customization](#).

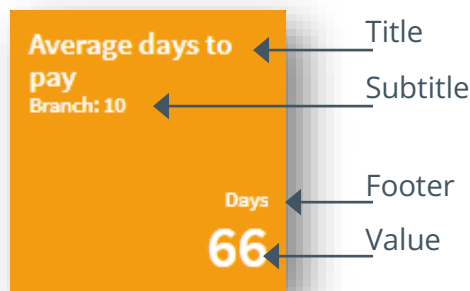
Insight Tile Customization

The previous topics have described how easy it is to create a new tile and add it to a user's workspace.

In this topic we are going to unpack some of the additional customizations that can be used to make the tile as effective as possible. Remember the purpose of an **Insight Tile** is to show concise information to the user allowing them to monitor key business values and act if needed.

TEXT TILE FORMATTING

A text tile consists of up to four components as shown below:



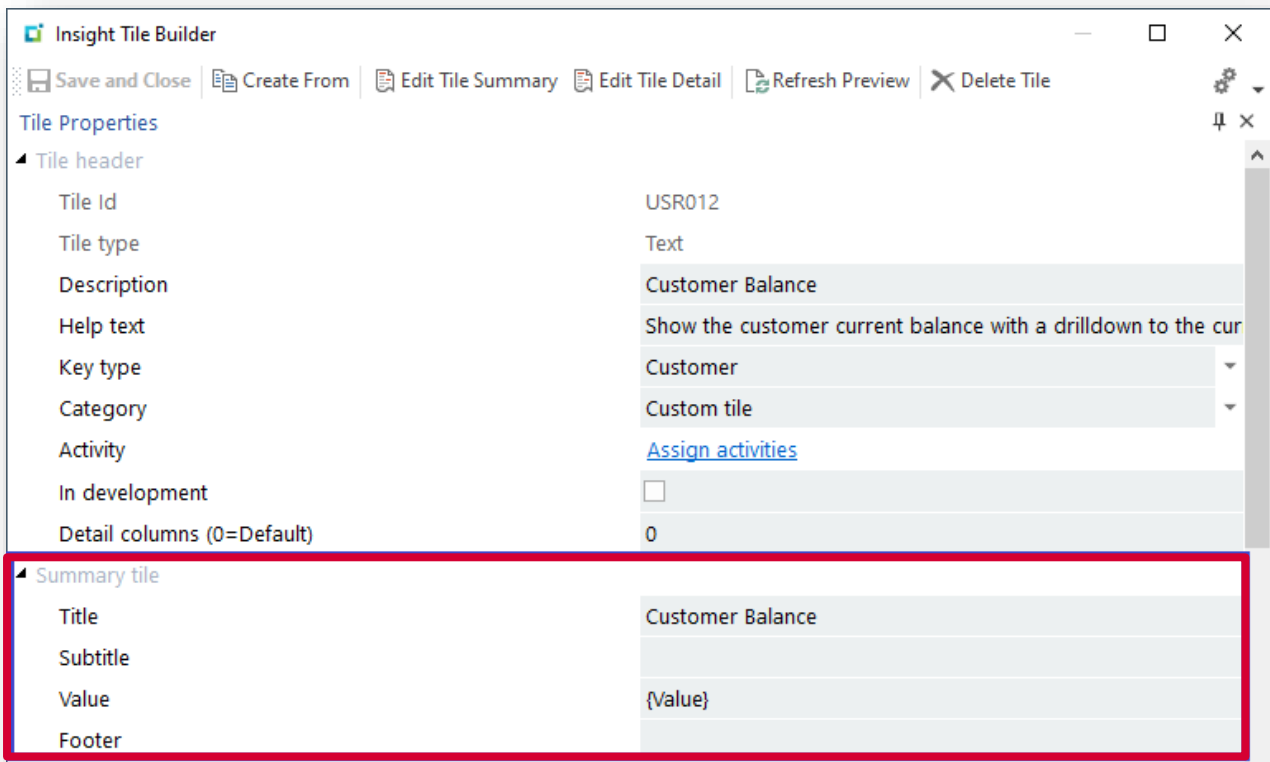
Note that the size, alignment, and font of each text component may vary depending on your SYSPRO release and theme selected. The colors can be defined when the tile is added to the user's workspace and modified based on a KPI threshold at run time.

Also note that in this example the Footer field is above the Value field.

The following table describes the text tile component attributes:

Component	Typical font size	Mandatory or Optional	SQL Column (exact case)
Title	Medium	Mandatory	{not applicable}
Subtitle	Smallest	Optional	SubTitle
Value	Largest	Mandatory	Value
Footer	Smallest	Optional	Footer

By default, the Insight Tile Builder defines the Summary tile information as shown below:



TEXT TILE - TITLE

The Title defaults to the tile description. In this example 'Customer Balance'.

However, a text tile is physically quite small, especially when selecting the small tile size when adding it to the user's workspace. You may want to override the tile title to be more concise.

Also remember that depending on where the tile is intended to be used, it may be clear that (for example) we are talking about a customer. In this case you could shorten the title to simply 'Balance'.

SUBTITLE, VALUE AND FOOTER

There are three additional parts to a text tile: Subtitle, Value and Footer. Value is mandatory and the other two are optional.

They all work in the same way and all support the concept of **receiving variables**, so have been grouped together for this explanation.

RECEIVING VARIABLES (SUBTITLE, VALUE AND FOOTER)

Any text in the summary tile entry fields will be shown on the tile 'as is'.

This can be useful to output, say, a unit of measure or other explanatory text.

For example, if calculating the average age of all invoices, you could calculate the average age and output that as the Value – and then the Footer could be hard coded to the phrase 'Days'.

You can also include **receiving variables** in the summary tile entry fields for Subtitle, Value, and Footer fields.

Receiving variables are strings surrounded with curly braces '{}'. The value within the curly braces must be supplied from the Summary SQL statement as a matching column name.

For example, if the **receiving variable** is **{Value}** then you must return a SQL column named 'Value' (exact case).

In addition to defining **receiving variables**, you can also provide editing information to indicate how the received column value is to be further edited or formatted.

For example, you can specify how many decimals to show, whether you want to apply smart auto-numbering or whether you use some of the standard SYSPRO editing functions for values such as cost, price, value, quantity, dates and so on.

To apply editing to a **receiving variable** you append a dot and then the editing keyword. Some common examples are provided below:

- Smart Auto-numbering: {Value.AutoNumber}
- Smart Auto-numbering with currency: {Value.AutoCurrency}
- Show as an integer: {Value.Integer}
- Show as a percentage: {Value.Percentage}
- Use SYSPRO's numeric editing: {Value.Number}
- Use SYSPRO's cost editing: {Value.Cost}
- Use SYSPRO's long date format: {Value.DateLong}
- Use SYSPRO's short date format: {Value.DateShort}

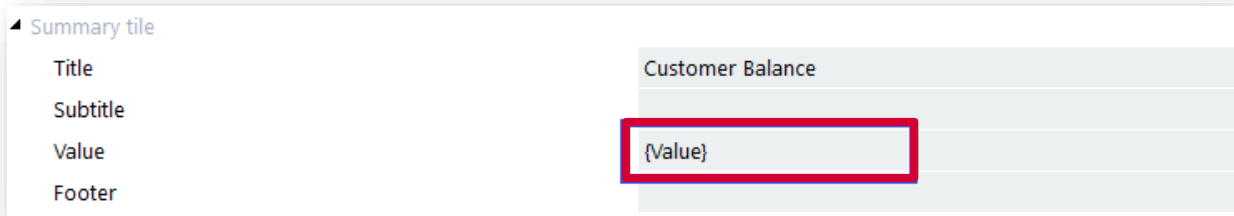
Some of these data types can be further modified using an additional parameter, often a number of decimal places or rounding modifier. See the following examples:

- Smart Auto-numbering with two decimals: {Value.AutoNumber.2}
- Smart Auto-numbering with currency with two decimals: {Value.AutoCurrency.2}
- Show as a percentage with two decimals: {Value.Percentage.2}

See [Appendix 2 - Data Types](#) for a complete list of available data types and where you can apply the number of decimals.

TEXT TILE DATA TYPE EXAMPLE - AUTOCURRENCY

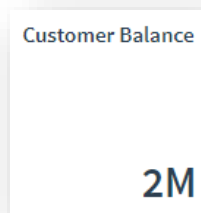
In previous examples we added a custom tile that showed the currently selected customer's balance. This used the default Summary tile properties for the 'Value' field **{Value}** – see below:



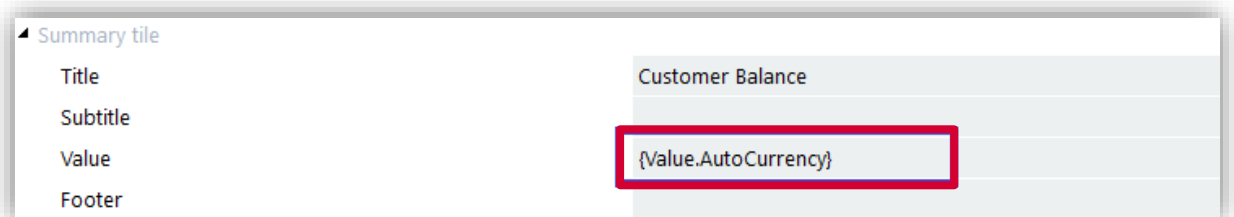
This means that the **receiving variable** named **{Value}** will be replaced by the SQL column named 'Value' (exact case).

Suppose that the actual customer balance was \$1,969,183.05 (just under \$2million). The default data type applied for numeric fields is **AutoNumber** with zero decimal places. See the topic: [Default Data types: Numeric - AutoNumber](#)

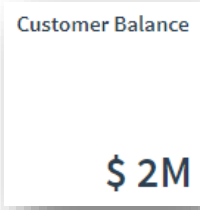
The tile will be rendered as shown below:



As this number represents a currency amount, as opposed to another type of number such as a cost, price, quantity etc., we are going to override the data type from the default of **AutoNumber** to the explicit data type: **AutoCurrency**.



The tile will now be rendered as shown below. In this case prefixing the value with the currency symbol '\$'.



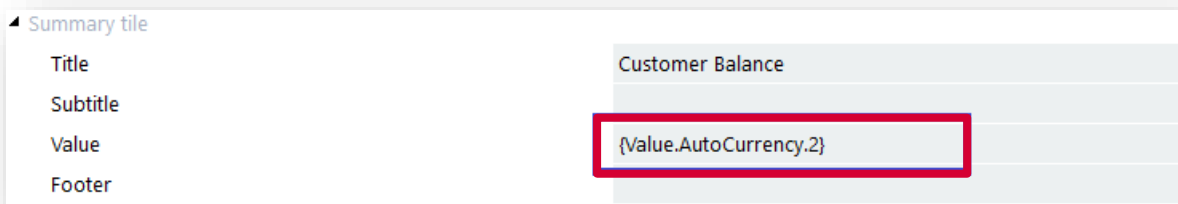
Customer Balance

\$ 2M

Note: If you are using an application (such as the Customer Query) and the tile definition is changed by another user (or another instance) then you will have to return to the menu and reload the application for the change to take effect. If the tile resides on the menu then you must exit SYSPRO and reload for the change to take effect.

Next, we are going to show this value rounded to the nearest two decimal places. If you use the default rounding to zero decimal places, the value '\$2M' could be any balance in the range '\$1.50M' thru '\$2.49M' – this is quite a wide range and could be misleading.

We append the rounding modifier '.2' to the **AutoCurrency** data type as shown below:



Summary tile	
Title	Customer Balance
Subtitle	
Value	{Value.AutoCurrency.2}
Footer	

The tile will then show the currency value rounded to the nearest two decimal places:



Customer Balance

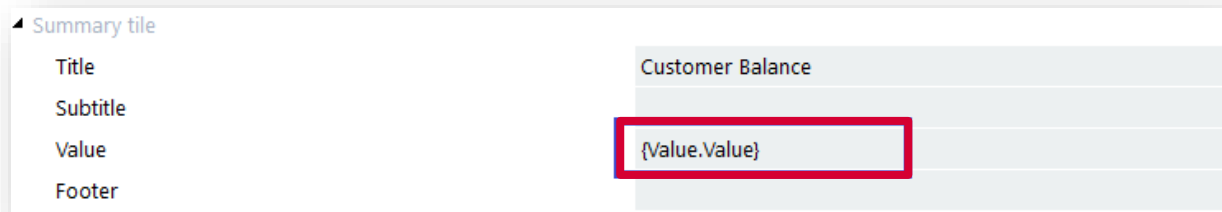
\$ 1.97M

This is much more effective as a simple text tile than the original one showing simply '2M'.

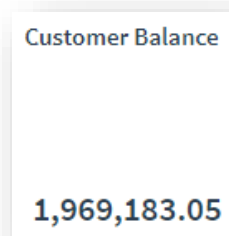
TEXT TILE DATA TYPE EXAMPLE - VALUE

Suppose we wish to show the customer balance to the full number of digits, including decimals – we could use the **Value** data type.

The **Value** data type uses the company setup options for editing numeric values. If configured, the operator override will also apply.



The tile will then be rendered as shown below:



Whilst showing the full customer balance detail can be useful, the summarized **AutoCurrency** data type is often more effective when used as a tile value – optionally using the rounding modifier.

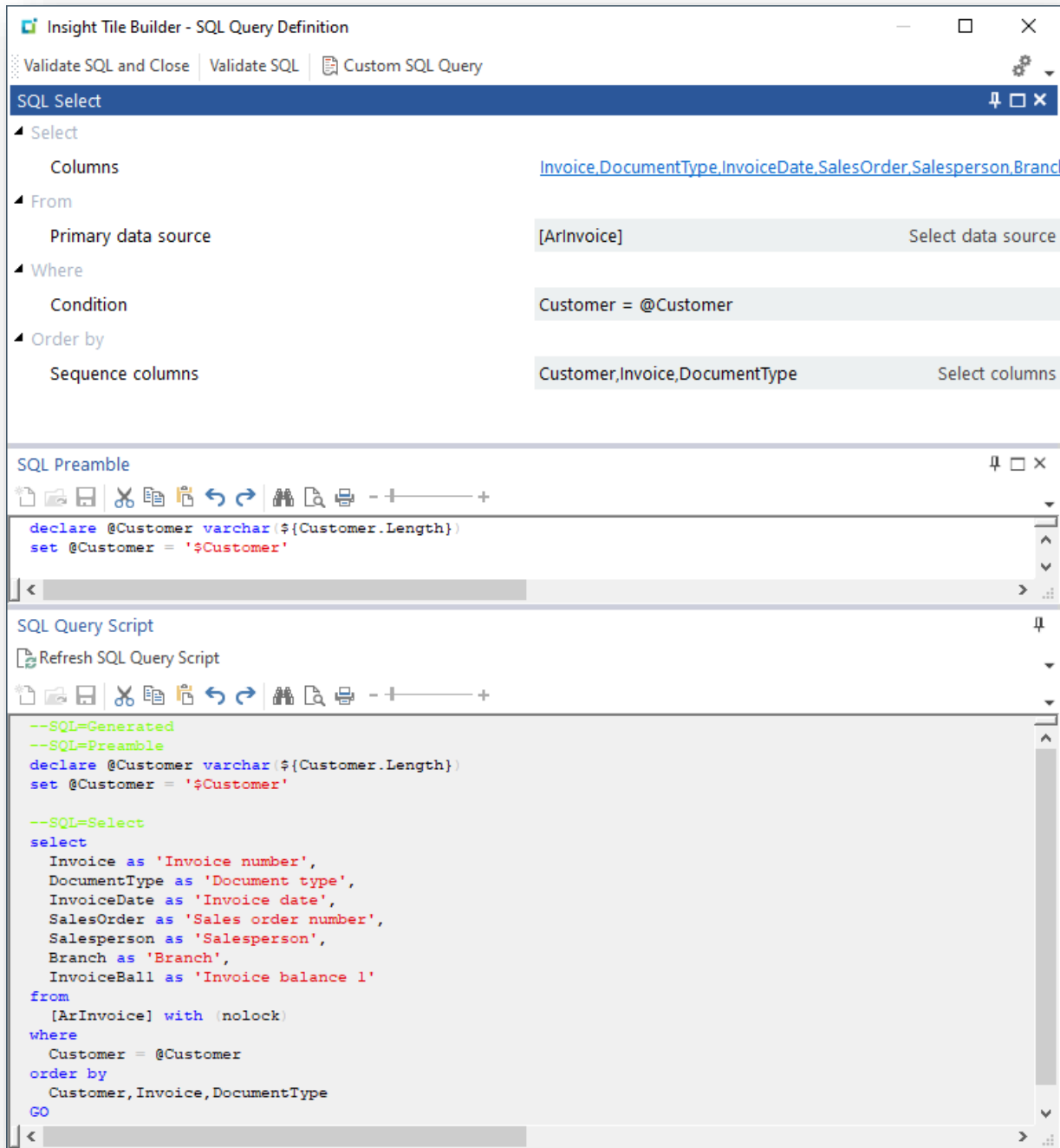
TEXT TILE DETAIL FORMATTING

When you click on a text tile to show the drilldown, your Detail SQL statement, either generated or custom, is used to show the detail in a list.

Each row returned from your SQL statement is shown as a row in the list and each column returned from your SQL statement is shown as a column in the list.

The column captions use the value of the 'as' clause from your SQL statement.

When generating your Detail SQL statement, the 'as' clause is generated from the selected column description from the SYSPRO data dictionary.



The example shown above will be rendered at run time as shown below:

Invoice number	Document type	Invoice date	Sales order number	Salesperson	Branch	
00000000100288	I	18/03/2013	000000000000650	100	10	
00000000100293	I	19/04/2013	000000000000655	100	10	
00000000100299	I	22/04/2013	000000000000662	100	10	
00000000100301	I	26/04/2013	000000000000662	100	10	
00000000100303	I	03/05/2013	000000000000663	100	10	
00000000100305	I	27/05/2013	000000000000655	100	10	
00000000100307	I	27/05/2013	000000000000663	100	10	
00000000100312	I	03/06/2013	000000000000668	100	10	
00000000100325	I	26/07/2013	000000000000672	100	10	
00000000100326	I	08/08/2013	000000000000676	100	10	
00000000100330	I	09/09/2013	000000000000679	100	10	
00000000100337	I	11/10/2013	000000000000685	100	10	
00000000100345	I	08/11/2013	000000000000691	100	10	
00000000100361	I	30/12/2013	000000000000697	100	10	

This is relatively easy to read and understand by a user. However, just like the summary tile information, we can further improve this output to make the user's experience even better.

TEXT TILE DETAIL – COLUMN CAPTION

SYSPRO has logic built-in to each list so that if it finds column captions that match pre-determined names, it understands the meaning of the data in that column and applies appropriate formatting (such as applying any rules for showing the column) and where relevant provide a smart link (hyperlink) to further drill into the cell value.

To override the caption, we change the Detail SQL definition from generated to custom. This is done by clicking on the 'Custom SQL Query' toolbar button. The SQL Query Script is then enabled allowing you to edit the previously generated SQL script.

In our example we are going to change the 'sales order number' column to say 'sales order'.

```

--SQL=Preamble
declare @Customer varchar(#{Customer.Length})
set @Customer = '#{Customer}'

--SQL=Select
select
    Invoice as 'Invoice number',
    DocumentType as 'Doc type',
    InvoiceDate as 'Invoice date',
    SalesOrder as 'Sales order',
    Salesperson as 'Salesperson',
    Branch as 'Branch',
    InvoiceBall as 'Balance'
from
    [ArInvoice] with (nolock)
where
    Customer = @Customer
order by
    Customer, Invoice, DocumentType
    
```

We have also shortened the caption against the DocumentType column to 'Doc type' and changed the caption against column InvoiceBal1 to 'Balance'.

The drilldown detail list now looks as follows:

Invoice number	Doc type	Invoice date	Sales order	Salesperson	Branch	Balance
00000000100377	I	13/03/2014	000715	100	10	0
00000000100391	I	11/04/2014	000724	100	10	0
00000000100394	I	09/05/2014	000730	100	10	0
00000000100405	I	10/06/2014	000739	100	10	0
00000000100413	I	18/07/2014	000745	100	10	0
00000000100415	I	06/08/2014	000746	100	10	0
00000000100421	I	25/09/2014	000750	100	10	0
00000000100431	I	27/10/2014	000759	100	10	0
00000000100441	I	18/11/2014	000768	100	10	0
00000000100452	I	22/12/2014	000778	100	10	0
00000000100461	I	16/01/2015	000787	100	10	0
00000000100466	I	09/02/2015	000793	100	10	448,000
00000000100476	I	05/03/2015	000803	100	10	914,000
00000000100485	I	29/03/2015	000793	100	10	0
00000000100497	I	01/04/2015	000815	100	10	584,800

Showing 41 item(s)

The Sales order column now shows the sales order numbers using the presentation format logic and provides a smart link.

Also, by shortening some of the caption names the invoice balance is now visible.

Note: When editing the Insight Tile Definition on a separate instance, you should save the Insight Tile on the one instance and you can immediately click on the detail drilldown in the other instance.

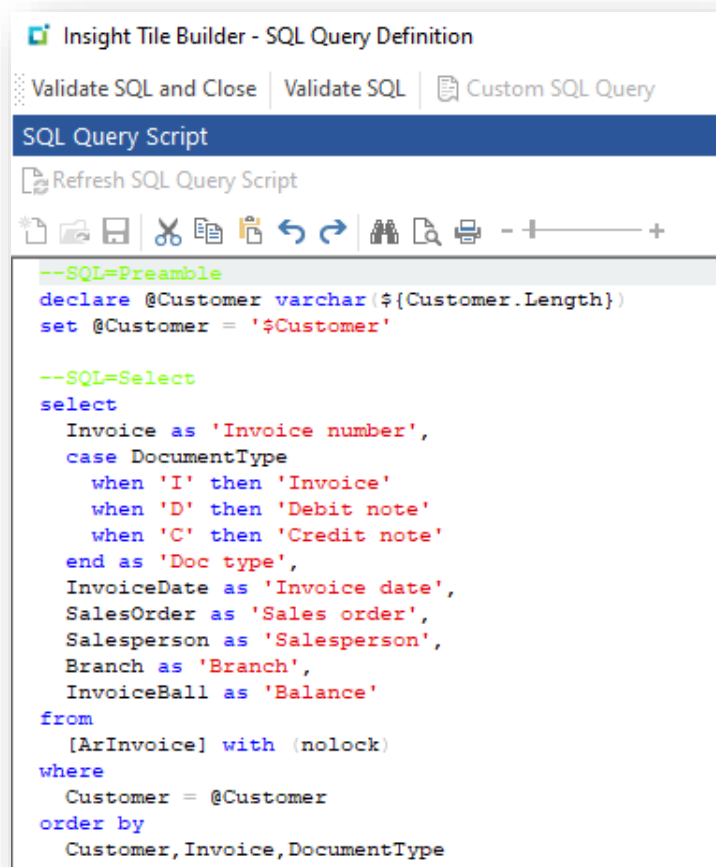
You may also find it useful to refer to the topic: [Detail SQL – Keeping the initial drilldown list simple](#)

TEXT TILE DETAIL – CASE STATEMENT

In the worked example we have a column named **DocumentType** that returns a flag indicating the document type – one of: 'I' for Invoice, 'D' for Debit note or 'C' for Credit note.

Showing a column value of 'I', 'D' or 'C' may be confusing to the user.

Therefore, we are going to customize the SQL statement to use a 'case' statement to return one of the document type descriptions – see the example below:



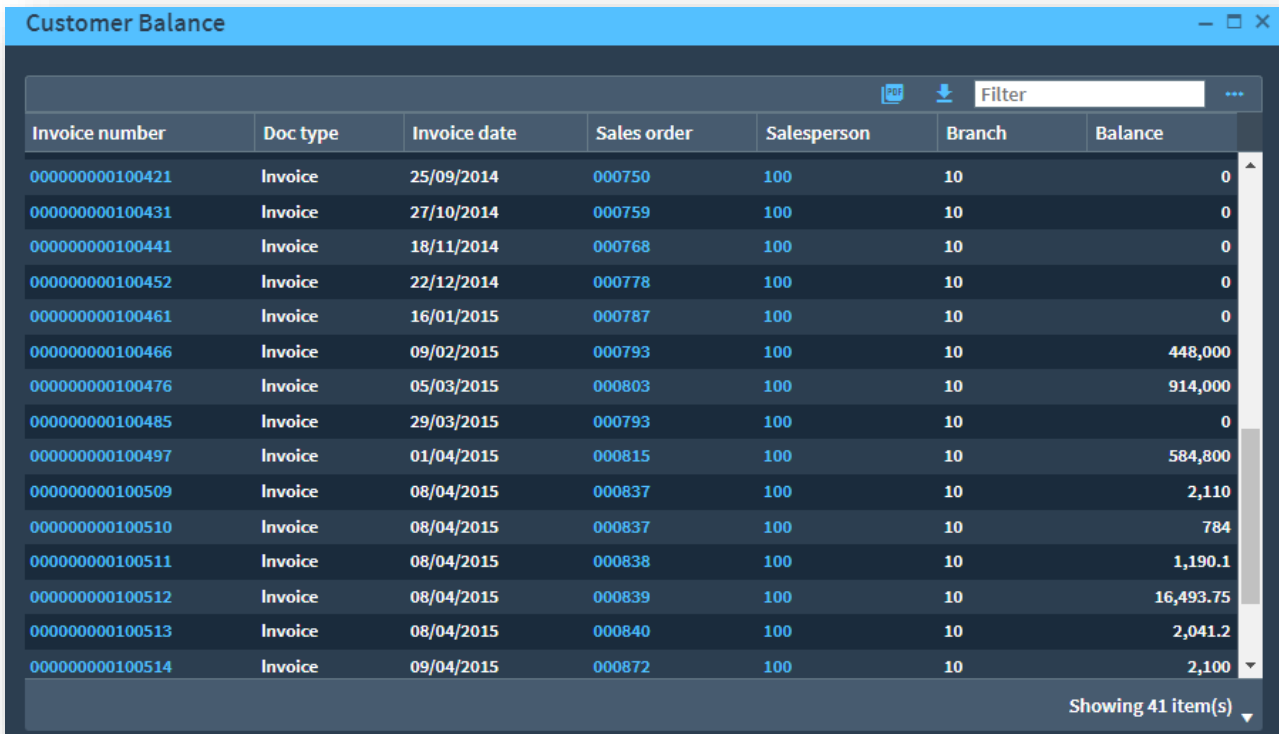
```
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '{Customer}'

--SQL=Select
select
  Invoice as 'Invoice number',
  case DocumentType
    when 'I' then 'Invoice'
    when 'D' then 'Debit note'
    when 'C' then 'Credit note'
  end as 'Doc type',
  InvoiceDate as 'Invoice date',
  SalesOrder as 'Sales order',
  Salesperson as 'Salesperson',
  Branch as 'Branch',
  InvoiceBall as 'Balance'
from
  [ArInvoice] with (nolock)
where
  Customer = @Customer
order by
  Customer, Invoice, DocumentType
```

The statement contains the following fragment:

```
case DocumentType
  when 'I' then 'Invoice'
  when 'D' then 'Debit note'
  when 'C' then 'Credit note'
end as 'Doc type',
```

At run time the drilldown list now shows:



Invoice number	Doc type	Invoice date	Sales order	Salesperson	Branch	Balance
00000000100421	Invoice	25/09/2014	000750	100	10	0
00000000100431	Invoice	27/10/2014	000759	100	10	0
00000000100441	Invoice	18/11/2014	000768	100	10	0
00000000100452	Invoice	22/12/2014	000778	100	10	0
00000000100461	Invoice	16/01/2015	000787	100	10	0
00000000100466	Invoice	09/02/2015	000793	100	10	448,000
00000000100476	Invoice	05/03/2015	000803	100	10	914,000
00000000100485	Invoice	29/03/2015	000793	100	10	0
00000000100497	Invoice	01/04/2015	000815	100	10	584,800
00000000100509	Invoice	08/04/2015	000837	100	10	2,110
00000000100510	Invoice	08/04/2015	000837	100	10	784
00000000100511	Invoice	08/04/2015	000838	100	10	1,190.1
00000000100512	Invoice	08/04/2015	000839	100	10	16,493.75
00000000100513	Invoice	08/04/2015	000840	100	10	2,041.2
00000000100514	Invoice	09/04/2015	000872	100	10	2,100

Notice in this example all documents returned are type: Invoice.

TEXT TILE DETAIL – FILTERING ROWS

In the worked example you may have noticed that the list is showing all invoices, even zero balance ones.

You may wish to filter out zero balance invoices by customizing the condition statement.

See the following detail SQL statement with a modified 'where' condition:

Insight Tile Builder - SQL Query Definition

Validate SQL and Close | Validate SQL | Custom SQL Query

SQL Query Script

Refresh SQL Query Script

```

--SQL=Preamble
declare @Customer varchar(#{Customer.Length})
set @Customer = '¢Customer'

--SQL=Select
select
    Invoice as 'Invoice number',
    case DocumentType
        when 'I' then 'Invoice'
        when 'D' then 'Debit note'
        when 'C' then 'Credit note'
    end as 'Doc type',
    InvoiceDate as 'Invoice date',
    SalesOrder as 'Sales order',
    Salesperson as 'Salesperson',
    Branch as 'Branch',
    InvoiceBall as 'Balance'
from
    [ArInvoice] with (nolock)
where
    Customer = @Customer
    and InvoiceBall <> 0
order by
    Customer, Invoice, DocumentType

```

At run time the drilldown list now shows:

Customer Balance

Filter

Invoice number	Doc type	Invoice date	Sales order	Salesperson	Branch	Balance
000000000100466	Invoice	09/02/2015	000793	100	10	448,000
000000000100476	Invoice	05/03/2015	000803	100	10	914,000
000000000100497	Invoice	01/04/2015	000815	100	10	584,800
000000000100509	Invoice	08/04/2015	000837	100	10	2,110
000000000100510	Invoice	08/04/2015	000837	100	10	784
000000000100511	Invoice	08/04/2015	000838	100	10	1,190.1
000000000100512	Invoice	08/04/2015	000839	100	10	16,493.75
000000000100513	Invoice	08/04/2015	000840	100	10	2,041.2
000000000100514	Invoice	09/04/2015	000872	100	10	2,100
000000000800055	Credit note	08/04/2015		100	10	-86
000000000900036	Debit note	08/04/2015		100	10	250
_CRD01	Invoice	31/03/2015		100	10	-2,500

Showing 12 item(s)

Note that there are only 12 invoices shown – previously all 41 invoices were shown, most of which were zero balance.

TEXT TILE DETAIL – COLUMN SEQUENCING

In the worked example we started with a generated detail SQL statement. This simply inserted the columns in the sequence that they appeared in the data dictionary.

You may want to re-sequence the columns to show the most important columns first.

You can simply edit the SQL statement and change the order of the columns. They are shown in the drilldown list in the order in which the columns are defined in the 'select' statement.

TEXT TILE DETAIL – COLUMN TYPE OVERRIDE

Sometimes you may wish to override the default drilldown list column editing to better suit the data being shown.

You apply the column type override fields by appending **{xxxx}** to the column caption. The following override phrases are supported:

- {String}
- {Guid}
- {Number.d}
- {Time}
- {Time.f}
- {Datm}

For example:

```
CurrentYear as 'Year{String}'
```

Important: The maximum length of the column caption (including any override) is 30 characters.

COLUMN TYPE OVERRIDE {STRING}

The **{String}** override forces the numeric value (typically an integer) to be returned and edited as if it were an alphanumeric field.

This prevents numeric editing rules, such as 1000 separators, being applied to the number.

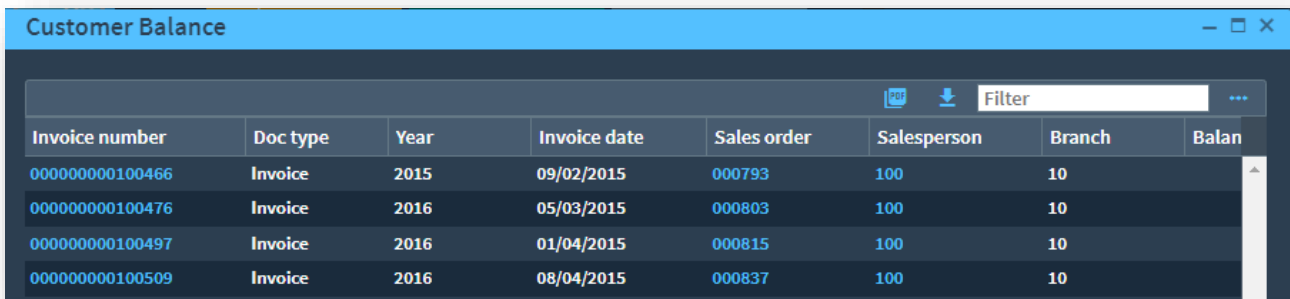
One usage could be when showing a column that represents a 4-digit year. If the year was (say) 2022, we do not want the year to be edited as 2,022.

For example, the column **InvoiceYear** in the **ArInvoice** table is described a decimal(4,0) and stores the 4-digit year in which the invoice was posted.

An example usage in this case could be:

```
InvoiceYear as 'Year{String}'
```

In the example below the Year column is shown as expected:



Invoice number	Doc type	Year	Invoice date	Sales order	Salesperson	Branch	Balan
00000000100466	Invoice	2015	09/02/2015	000793	100	10	
00000000100476	Invoice	2016	05/03/2015	000803	100	10	
00000000100497	Invoice	2016	01/04/2015	000815	100	10	
00000000100509	Invoice	2016	08/04/2015	000837	100	10	

COLUMN TYPE OVERRIDE {GUID}

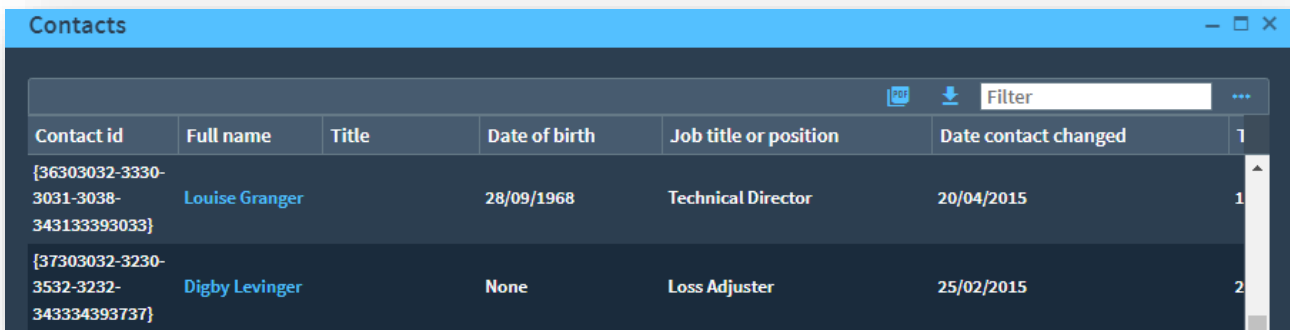
The **{Guid}** override forces the column to be shown using the classic GUID format.

For example, the column **ContactId** in the **CrnContact** table is described a **uniqueidentifier** 64-bit GUID uniquely identifying the contact.

An example usage in this case could be:

```
ContactId as 'Contact id{Guid}'
```

In the example below the Contact id column is shown as expected:



Contact id	Full name	Title	Date of birth	Job title or position	Date contact changed	
{36303032-3330-3031-3038-343133393033}	Louise Granger		28/09/1968	Technical Director	20/04/2015	1
{37303032-3230-3532-3232-343334393737}	Digby Levinger		None	Loss Adjuster	25/02/2015	2

It should be mentioned that typically showing GUIDs to a user is not a good user experience. This example was just provided to demonstrate the edit override capability.

COLUMN TYPE OVERRIDE {NUMBER.D}

The **{Number.d}** override forces the column to be shown as a numeric field with the number of decimals 'd' (where 'd' is a number from 0 to 6).

By default, most numbers are edited to the number of decimals with trailing zeros removed.

However, in some cases it's more effective to force a specific number of decimals so that the numbers are aligned more appropriately.

For example, the column **InvoiceBal1** in the **ArInvoice** table has been used in examples and we wish to force two decimal places.

An example usage in this case could be:

```
InvoiceBal1 as 'Balance{Number.2}'
```

In the example below the Balance column values are right aligned with two decimal places:

Invoice number	Doc type	Invoice date	Sales order	Salesperson	Branch	Balance
00000000100466	Invoice	09/02/2015	000793	100	10	448,000.00
00000000100476	Invoice	05/03/2015	000803	100	10	914,000.00
00000000100497	Invoice	01/04/2015	000815	100	10	584,800.00
00000000100509	Invoice	08/04/2015	000837	100	10	2,110.00
00000000100510	Invoice	08/04/2015	000837	100	10	784.00
00000000100511	Invoice	08/04/2015	000838	100	10	1,190.10

COLUMN TYPE OVERRIDE {TIME.F}

Historically, many columns in SYSPRO tables that represented times were stored as an 8-digit numeric field (decimal(8,0)).

These times represent HHMMSSFF where

- HH is a 2-digit hour (00-23)
- MM is a 2-digit minute (00-59)
- SS is a 2-digit second (00-59)
- FF is a 2-digit fraction of a second (00-99)

The **{Time.f}** override forces the numeric column to be shown as an edited time in one of four formats. This is output as alphanumeric string.

Override	Format	Example
{Time.1}	HH	10
{Time.2}	HH:MM	10:15
{Time.3}	HH:MM:SS	10:15:45
{Time.4}	HH:MM:SS:FF	10:15:45:98

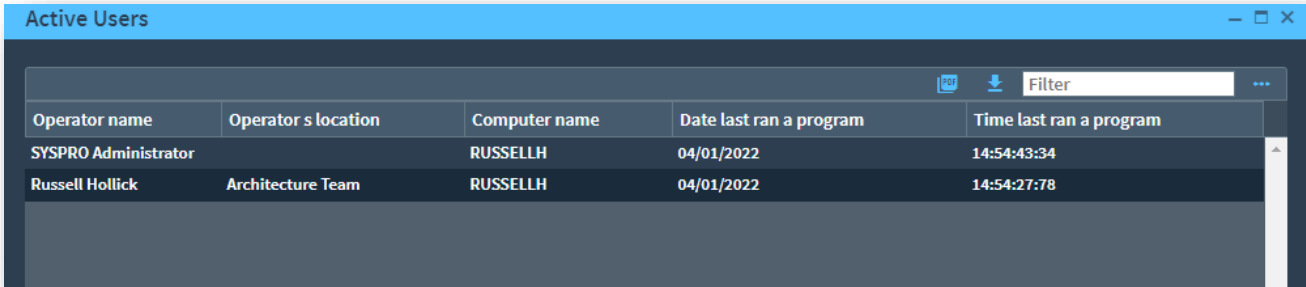
The simplified override **{Time}** is the same as **{Time.4}**.

For example, the column **LastUseTime** in the **AdmCurrentUsers** table represents the time that someone last ran a program.

An example usage in this case could be:

```
LastUseTime as 'Time last ran a program{Time}'
```

In the example below the Time last ran a program column is shown as an edited time.



Operator name	Operator's location	Computer name	Date last ran a program	Time last ran a program
SYSPRO Administrator		RUSSELLH	04/01/2022	14:54:43:34
Russell Hollick	Architecture Team	RUSSELLH	04/01/2022	14:54:27:78

COLUMN TYPE OVERRIDE {Datm}

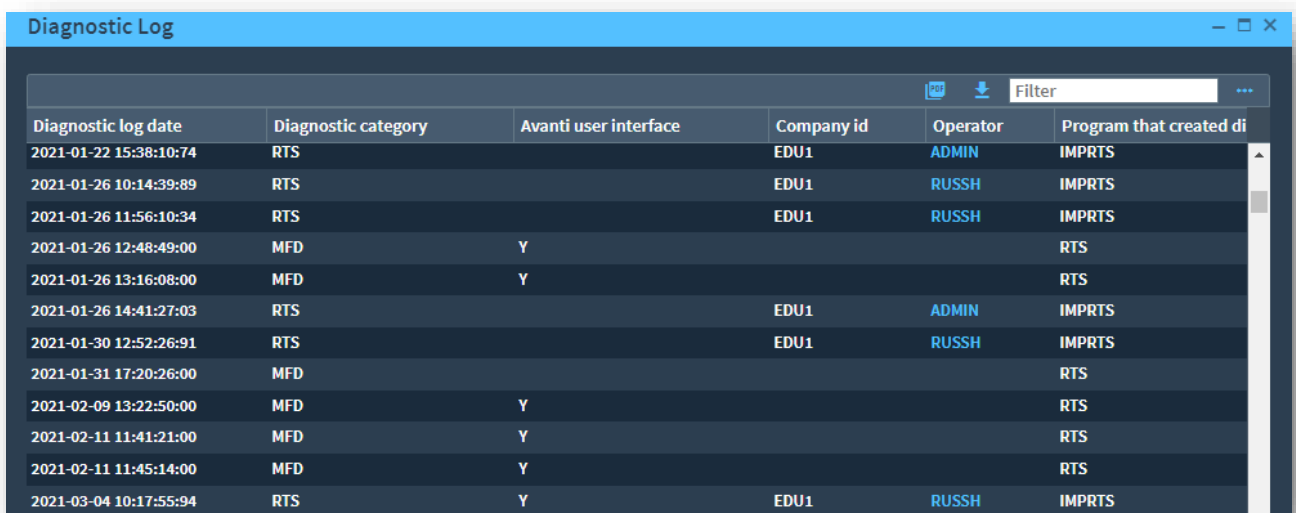
The {Datm} override forces a datetime column to be shown as YYYY-MM-DD HH:MM:SS:FF

For example, the column **LogDate** in the **AdmDiagSummary** table is defined as a SQL datetime data type.

An example usage in this case could be:

```
LogDate as 'Diagnostic log date{Datm}'
```

In the example below the Diagnostic log date column is shown as an edited datetime.

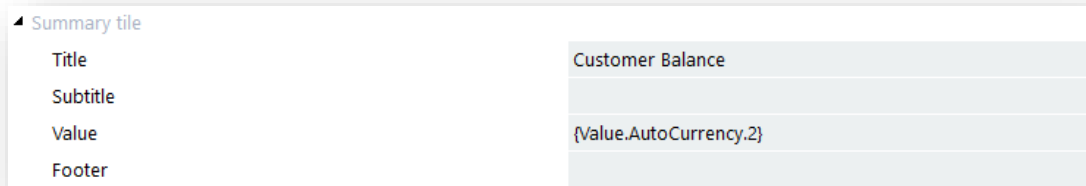


Diagnostic log date	Diagnostic category	Avanti user interface	Company id	Operator	Program that created di
2021-01-22 15:38:10:74	RTS		EDU1	ADMIN	IMPRTS
2021-01-26 10:14:39:89	RTS		EDU1	RUSSH	IMPRTS
2021-01-26 11:56:10:34	RTS		EDU1	RUSSH	IMPRTS
2021-01-26 12:48:49:00	MFD	Y			RTS
2021-01-26 13:16:08:00	MFD	Y			RTS
2021-01-26 14:41:27:03	RTS		EDU1	ADMIN	IMPRTS
2021-01-30 12:52:26:91	RTS		EDU1	RUSSH	IMPRTS
2021-01-31 17:20:26:00	MFD				RTS
2021-02-09 13:22:50:00	MFD	Y			RTS
2021-02-11 11:41:21:00	MFD	Y			RTS
2021-02-11 11:45:14:00	MFD	Y			RTS
2021-03-04 10:17:55:94	RTS	Y	EDU1	RUSSH	IMPRTS

Smart editing - AutoCurrency

When you configure a summary text tile that shows a currency value (such as the customer current balance) you should use the **AutoCurrency** data type.

For example:



Title	Customer Balance
Subtitle	
Value	{Value.AutoCurrency.2}
Footer	

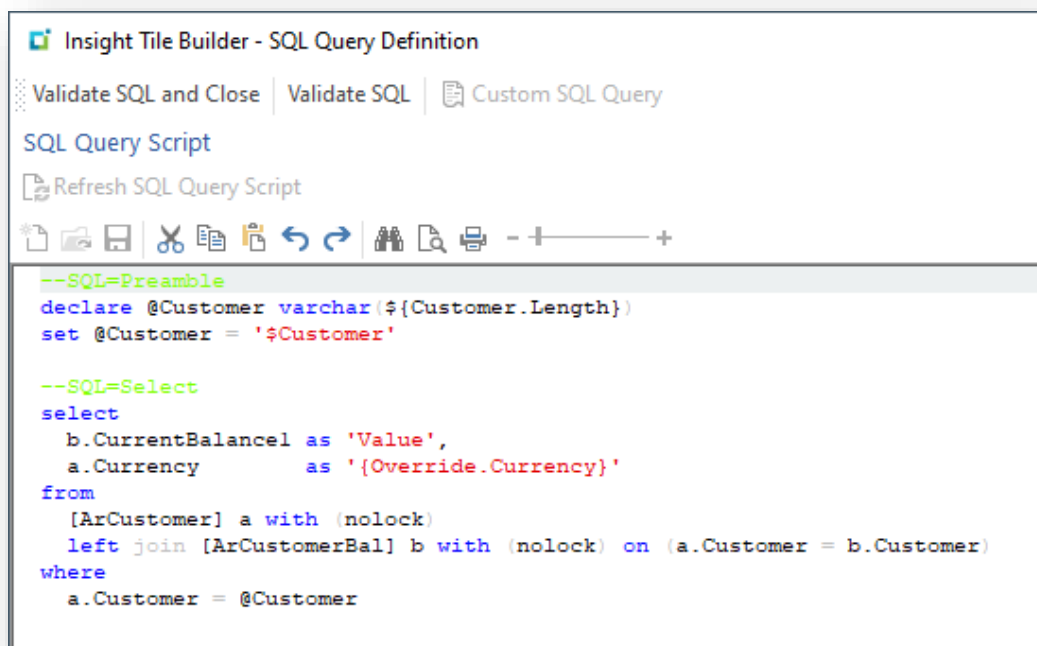
In this case we are rounding to two decimal places.

However, the default currency code that is used when using the **AutoCurrency** data type is the company default currency code. If you only have a single currency across all customers, then this is ok.

However, if you have multiple currencies then you will need to use the Customer's currency code. This is done by outputting the currency code to a column with the caption **{Override.Currency}**

This is a special caption that allows you to configure the currency code to be used when the **AutoCurrency** data type logic is expanded.

In the following example we have edited the custom Customer balance tile that had a generated summary SQL statement and customized it as shown below:



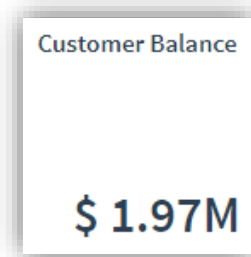
```
--SQL=Preamble
declare @Customer varchar(#{Customer.Length})
set @Customer = '#{Customer}'

--SQL=Select
select
  b.CurrentBalance1 as 'Value',
  a.Currency        as '{Override.Currency}'
from
  [ArCustomer] a with (nolock)
  left join [ArCustomerBal] b with (nolock) on (a.Customer = b.Customer)
where
  a.Customer = @Customer
```

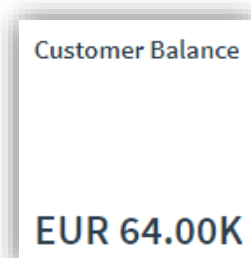

There are a few areas to note:

- The Customer balance resides on the table **ArCustomerBal** and the currency code on the customer master table **ArCustomer**. Therefore, we had to create a custom SQL definition and reference both tables with an appropriate join.
- We have selected the column **a.Currency** and output it with the special caption **{Override.Currency}**.

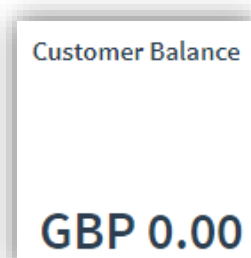
At run time if we query a customer with the currency code '\$' it shows:



If we then select a customer with the currency code 'EUR' it shows:



If we then select a customer with the currency code 'GBP' it shows:



You can also override the language code used to determine the currency abbreviation logic by using the **{Override.Language}** column caption.

See topic: [Appendix 4 – AutoNumber and AutoCurrency](#)

Parameters

In some cases, you may wish to add **Insight Tiles** to user's workspaces that only differ by one or more selection criteria – such as a different branch, area, class, category, salesperson, or other filter.

You could create a single **Insight Tile** for each of the required selection criteria, however, this would potentially lead to many tiles that only differ by one small change in the generated or customized SQL – often the conditional statement.

There are a few practical challenges with this approach:

- If the criterion is a single key field such as a branch, and you have hundreds of branches, then you would have to create hundreds of **Insight Tiles** – this is not a practical approach
- It's worse if the criteria consist of two or more fields such as branch, area, and class – the potential combinations are very large
- If you later wish to change all the similar tiles for some reason, you must edit each one individually
- If you wish to apply a KPI, then you must duplicate the KPI definition, one per tile

The solution to these challenges is for **Insight Tiles** to support the concept of parameters.

When you add a tile to a user's workspace, one or more parameters can be prompted. You can retain the default value or make an appropriate entry or selection. Each parameter value is persisted along with the tile definition against the user's workspace.

At run-time, the persisted parameter values are passed to the tile. You can use the parameter values in the SQL statement to filter or change the data selected as required.

PARAMETERS – WORKED EXAMPLE

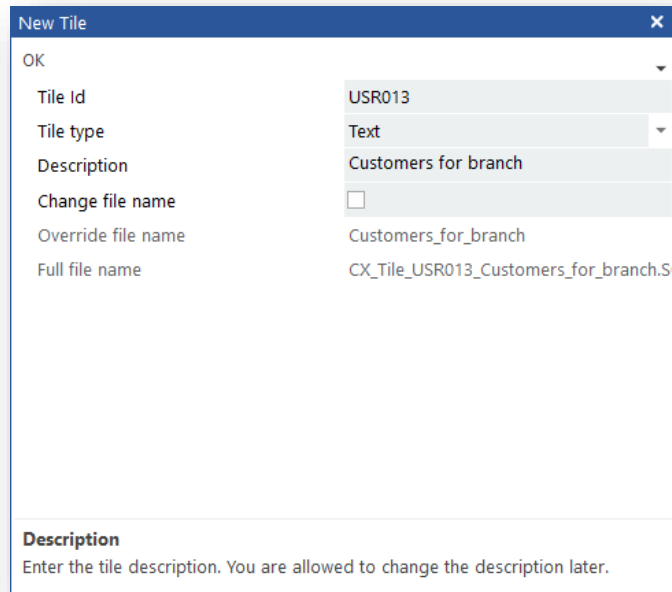
We will create a new custom tile that shows the number of customers assigned to a specific branch and show the tile on the menu. When you click on the tile, the list of customers that are assigned to the branch will be shown.

The branch code will be used as the parameter for filtering purposes.

We will add multiple instances of this tile to the same user's workspace. Each instance selecting a separate branch code.

Start the process by loading the Insight Tile Definition (IMPKPI) program and selecting 'New Tile'.

When the New Tile dialog shows, we're going to retain the generated tile id and enter the tile description.



New Tile

OK

Tile Id: USR013

Tile type: Text

Description: Customers for branch

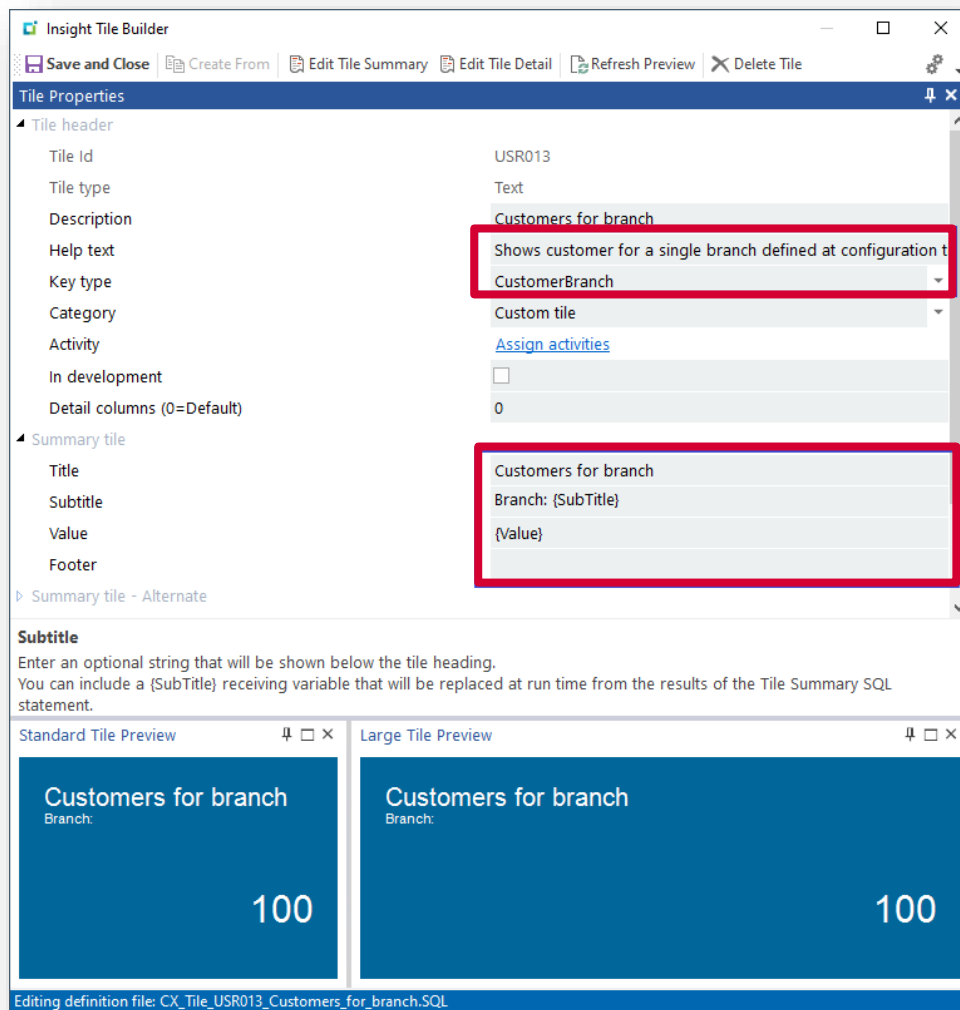
Change file name:

Override file name: Customers_for_branch

Full file name: CX_Tile_USR013_Customers_for_branch.S

Description
Enter the tile description. You are allowed to change the description later.

It is a good practice to expand on the tile description by entering relevant help text.



Insight Tile Builder

Save and Close | Create From | Edit Tile Summary | Edit Tile Detail | Refresh Preview | Delete Tile

Tile Properties

- Tile header
 - Tile Id: USR013
 - Tile type: Text
 - Description: Customers for branch
 - Help text: Shows customer for a single branch defined at configuration t
 - Key type: CustomerBranch
 - Category: Custom tile
 - Activity: [Assign activities](#)
 - In development:
 - Detail columns (0=Default): 0
- Summary tile
 - Title: Customers for branch
 - Subtitle: Branch: {SubTitle}
 - Value: (Value)
 - Footer:

Summary tile - Alternate

Subtitle
Enter an optional string that will be shown below the tile heading. You can include a {SubTitle} receiving variable that will be replaced at run time from the results of the Tile Summary SQL statement.

Standard Tile Preview | Large Tile Preview

Customers for branch
Branch:
100

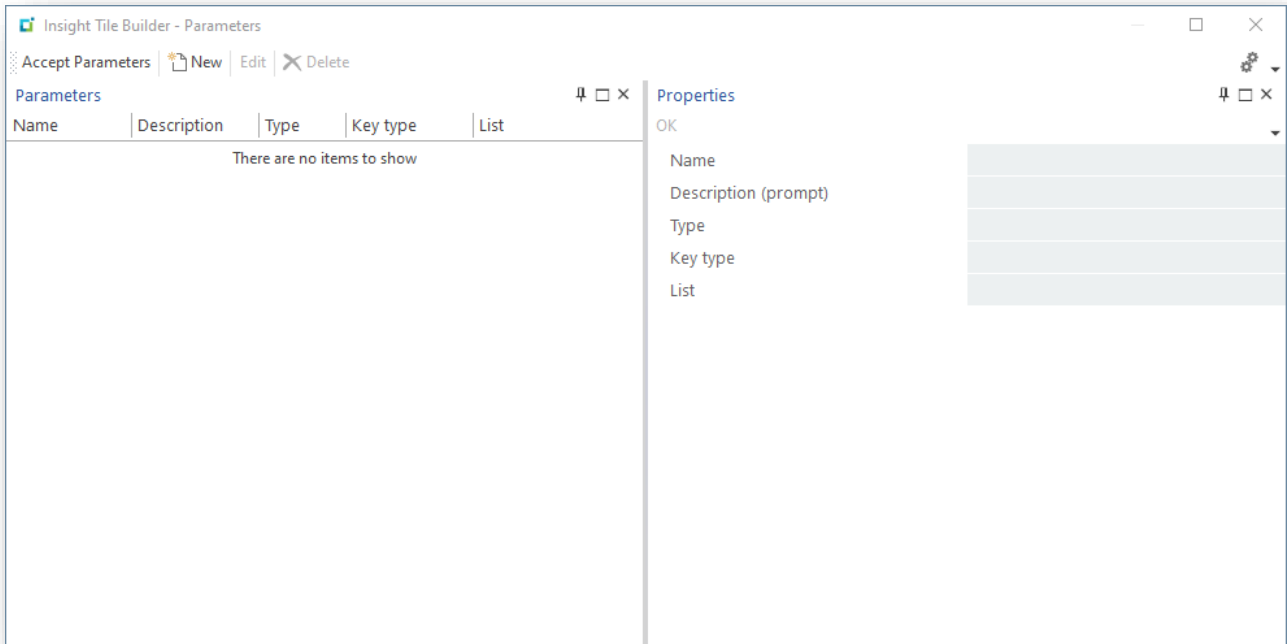
Customers for branch
Branch:
100

Editing definition file: CX_Tile_USR013_Customers_for_branch.SQL

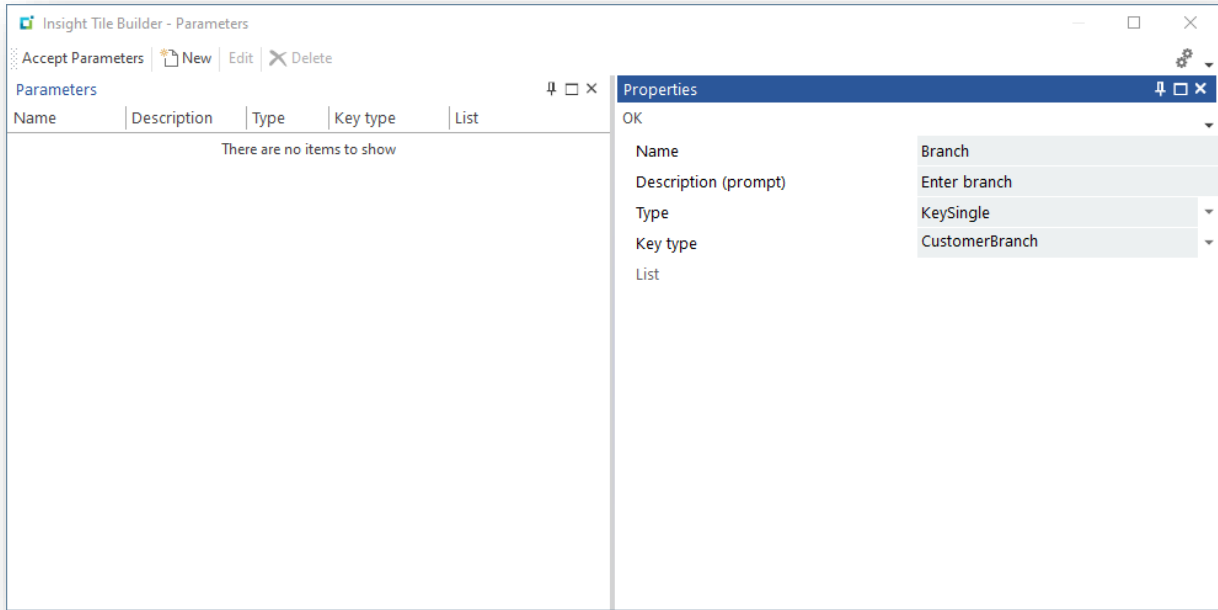
Considerations:

- Key type: It is recommended that if you are referencing a key field that you select it at the 'Key type' prompt – the main reason is that it will provide a more appropriate default Summary and Detail SQL – this will be useful later
- Title is retained from the tile description (change as required)
- Subtitle contains the plain text 'Branch: ' immediately followed by the **Receiving variable {SubTitle}**
 - The **Receiving variable** will be replaced at run time with the branch code
 - This allows the user to clearly see which branch has been selected
- The **Receiving variable {Value}** will be replaced by the number of customers that are assigned to the selected branch.

Now we will define the parameter by clicking on the **Assign parameters** hyperlink against the Configure parameters field. You will be presented with the **Insight Tile Builder – Parameters** dialog.



Select the 'New' toolbar button and enter a parameter as shown:

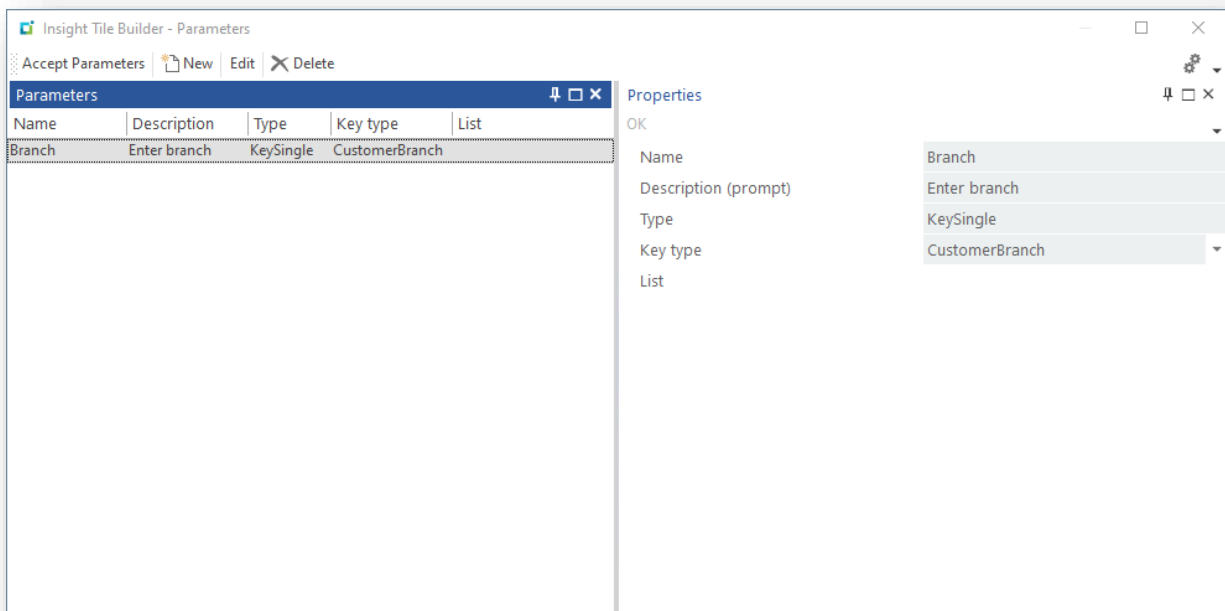


The Description will become the prompt when the tile is assigned to the user's workspace.

The type **KeySingle** informs the tile system that the prompt represents a single key – in this case the **CustomerBranch**.

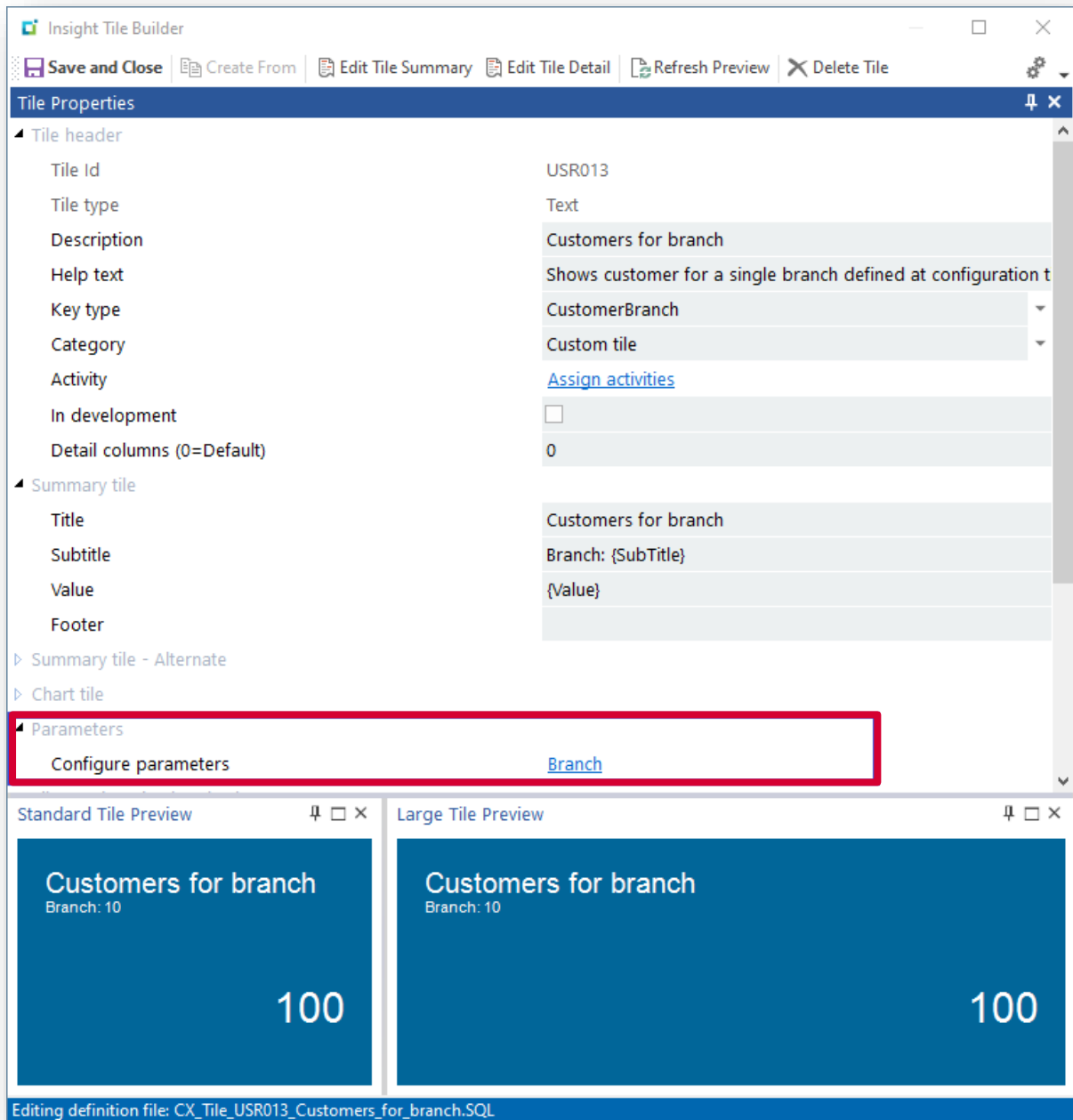
There other types of parameters – these will be covered later in the topic [Parameter types](#).

Then click 'OK' or press Enter. The parameter will be added to the list on the left allowing you to add another parameter or view existing parameters:



Click 'Accept Parameters' to return to the main Insight Tile Builder dialog.

Note that the Configure parameters field is updated to show a hyperlinked list of parameters. In this case the single parameter named 'Branch'.



We have also improved the tile preview simulated values to make it more realistic:

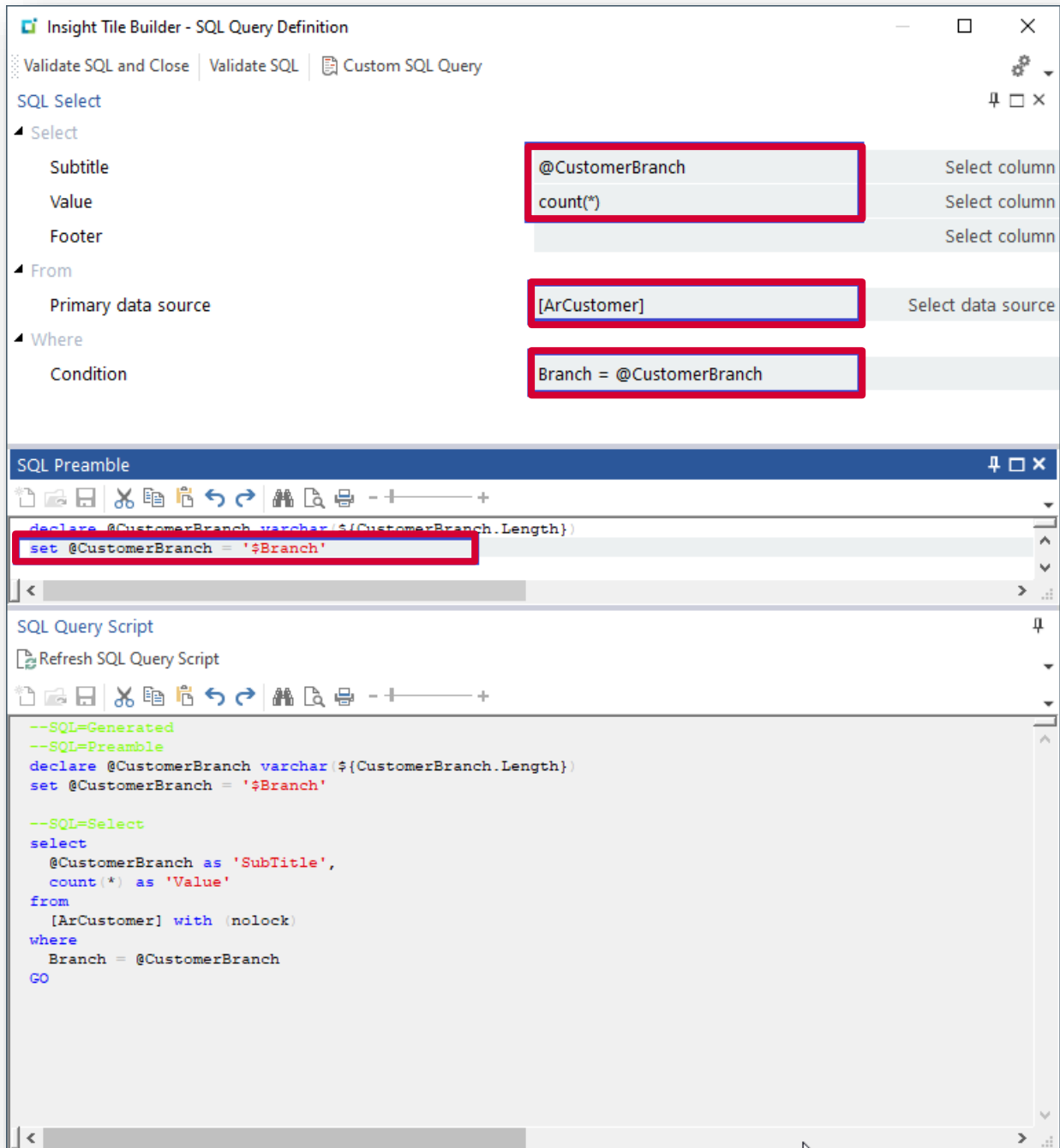
The screenshot displays the 'Tile preview simulated values' configuration panel. A red box highlights the 'Preview variable: {SubTitle}' field set to '10' and the 'Preview variable: {Value}' field set to '175'. Other settings include 'Preview variable: {Footer}', 'Foreground color' (White), 'Foreground color RGB' (255; 255; 255), 'Background color' (Primary), and 'Background color RGB' (1; 102; 153). Below the configuration is a text input for 'Preview variable: {Footer}' with instructions. At the bottom, two preview windows are shown: 'Standard Tile Preview' and 'Large Tile Preview', both displaying 'Customers for branch' with 'Branch: 10' and a large '175' value. The status bar at the bottom indicates the file being edited: 'CX_Tile_USR013_Customers_for_branch.SQL'.

Note the 'Preview variable {SubTitle}' field is set to '10' and the 'Preview variable {Value}' field is set to '175'.

These are just examples – you can adjust values to match your environment more closely.

We are now ready to use the Edit Tile Summary toolbar function.

We will be using the generated SQL dialog as follows:

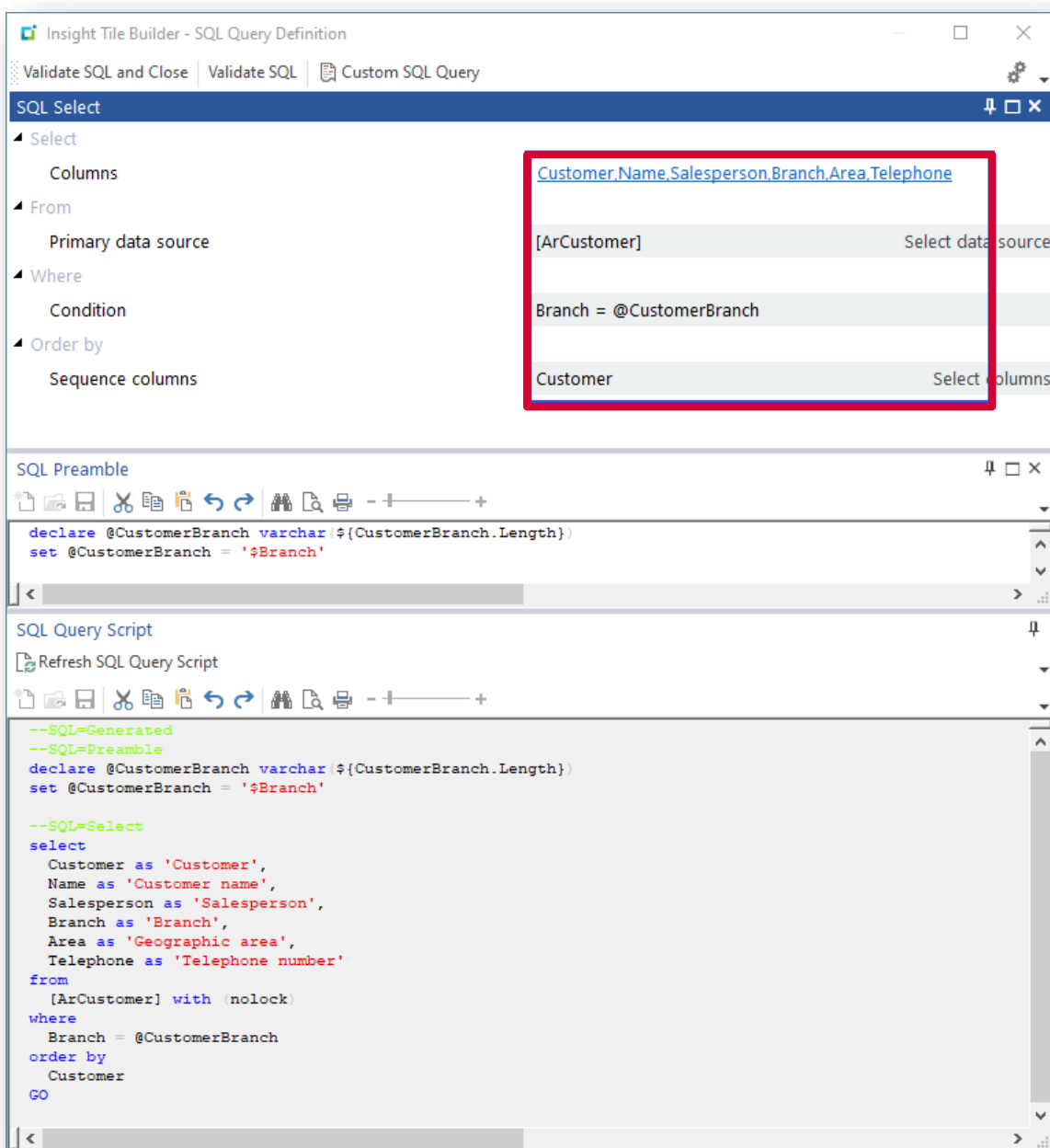


Considerations:

- Select the base table **ArCustomer** as the primary data source
- Enter **count(*)** against the Value prompt
 - This indicates that we wish to count the number of rows returned by the SQL select statement
- Enter **@CustomerBranch** against the Subtitle prompt
 - As we are aggregating (in our case counting) the number of rows we should return the variable that was used for the filter.

- The initial generated condition used a column named **CustomerBranch**. However, in the **ArCustomer** table the column name is **Branch**. You will need to edit this manually as shown above.
- Adjust the SQL Preamble to use the name of the parameter **\$Branch**
 - The default generated preamble referenced **\$CustomerBranch** but we named our parameter 'Branch'.
 - If we had named our parameter 'CustomerBranch' this would change would not have been necessary – it has been shown to demonstrate that you may need to adjust the SQL Preamble in some cases.

Next, we will define the drilldown detail. Select the Edit Tile Detail toolbar button and make the following entries.



You will note that the primary data source will be pre-populated with the same value as the summary SQL. Also, the condition and SQL Preamble will also be transferred.

Then select the columns you wish to show – in this case we selected:

- Customer
- Name
- Salesperson
- Branch
- Area
- Telephone

We are now ready to test this tile definition by adding it to a user’s workspace.

- Load the workspace editor (in our example we are using the menu)
- Select to add a tile
- Select the Category ('Custom tile') and the tile 'Customers for branch'

The Add tile dialog is shown below:

Add tile

1 Select tile to add:

Search for a tile...

< All / Custom tile

- Active sales orders
- Customer Balance
- Customers for branch**
- Days since last sale for company
- Default bar chart
- Default line chart

2 Configure tile:

Tile width

Small Large

Enter branch

10

Main text colour

Tile background colour

Clear colour selections

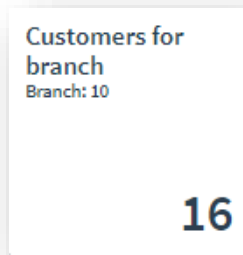
Add an icon →

Cancel Add tile

You will see that our parameter is being prompted – see 'Enter branch'.

We're going to enter a branch code of '10'.

When you save the workspace... in our test system this shows the following tile:



If you click on the tile to show the drilldown detail it will show the list of customers that are assigned to the branch. For example:

A screenshot of a data table window titled 'Customers for branch'. The table has columns for Customer, Customer name, Salesperson, Branch, Geographic area, and Telephone number. It lists six customer records.

Customer	Customer name	Salesperson	Branch	Geographic area	Telephone number
0000001	Bayside Bikes	100	10	N	206-555-4562
0000002	Bikes & Blades - North	100	10	N	416-555-8256
0000005	Cash Sales	103	10	N	
0000008	Garden and Sports	102	10	N	604-555-1523
0000009	Grand Adventures	101	10	O	
0000010	Maniac Sports - North	100	10	N	847-555-8756

To demonstrate the value of using the technique of using parameters, rather than hard-coding multiple similar tiles, we can now immediately add another instance of this tile, this time we will select branch '20'.

The user's menu could look as follows:



The drilldown on the tile for branch '20' could look as follows:

Customer	Customer name	Salesperson	Branch	Geographic area	Telephone number
0000003	Bikes & Blades - South	201	20	S	011-555-4262
0000007	Country Gardens - South	201	20	S	021-555-7485
0000011	Maniac Sports - South	201	20	S	334-555-9855
0000013	Out of Africa	200	20	S	011-555-2123
0000014	The Garden of Eden	200	20	S	031-555-8723
0000019	The Garden Executive	200	20	S	612-555-1092

It would be a simple case to add any number of instances of this tile selecting a branch as required.

PARAMETER TYPES

There are three types of parameters supported – they are:

- KeySingle
- KeyList
- List

PARAMETER TYPE – KEYSINGLE

When defining a parameter type **KeySingle**, you can select the key type from one of the standard key types.

The Key type is not currently used by the tile system but serves as a reference for the key type.

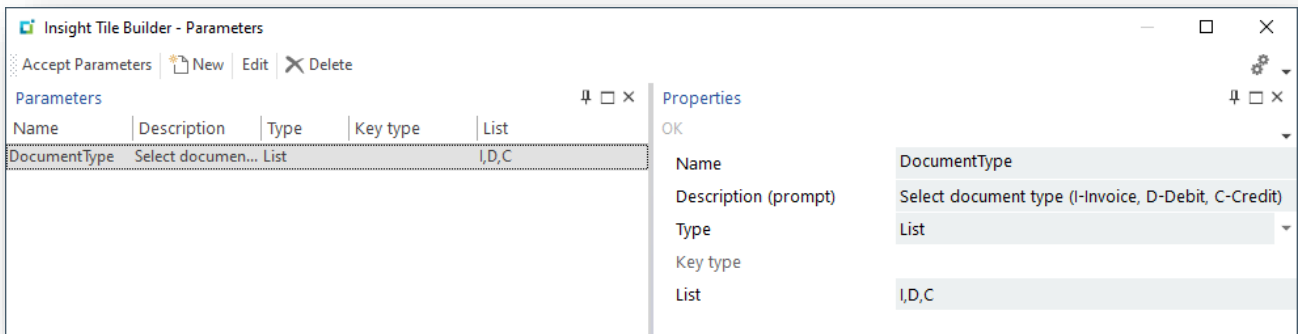
If you just want to allow a free-format alpha string to be entered at the time the tile is added to the user's workspace, then select **KeySingle** and leave the key type as spaces.

The topic [Parameters - Worked Example](#) earlier in this document covers this topic in more detail.

PARAMETER TYPE – LIST

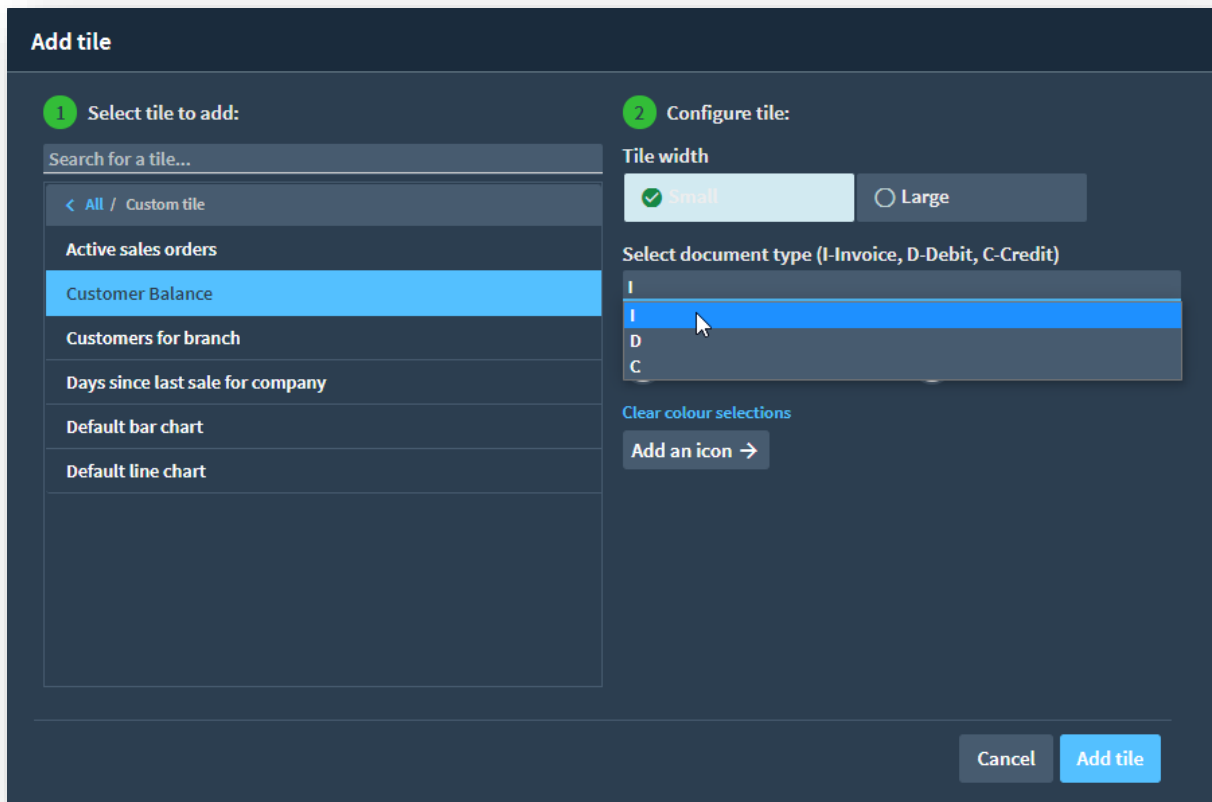
When defining a parameter type **List**, you can configure a comma separated list of items that can be selected at the time you add the tile to the user's workspace. The first entry will be the default value.

For example, if you wish to have a document type selection where: 'I' means Invoice, 'D' means Debit note and 'C' means Credit note - then you can define a parameter as shown below:



When adding the tile to the user's workspace you will be prompted for the document type. The default will be the first entry ('I') and you can select from the other entries in a dropdown list.

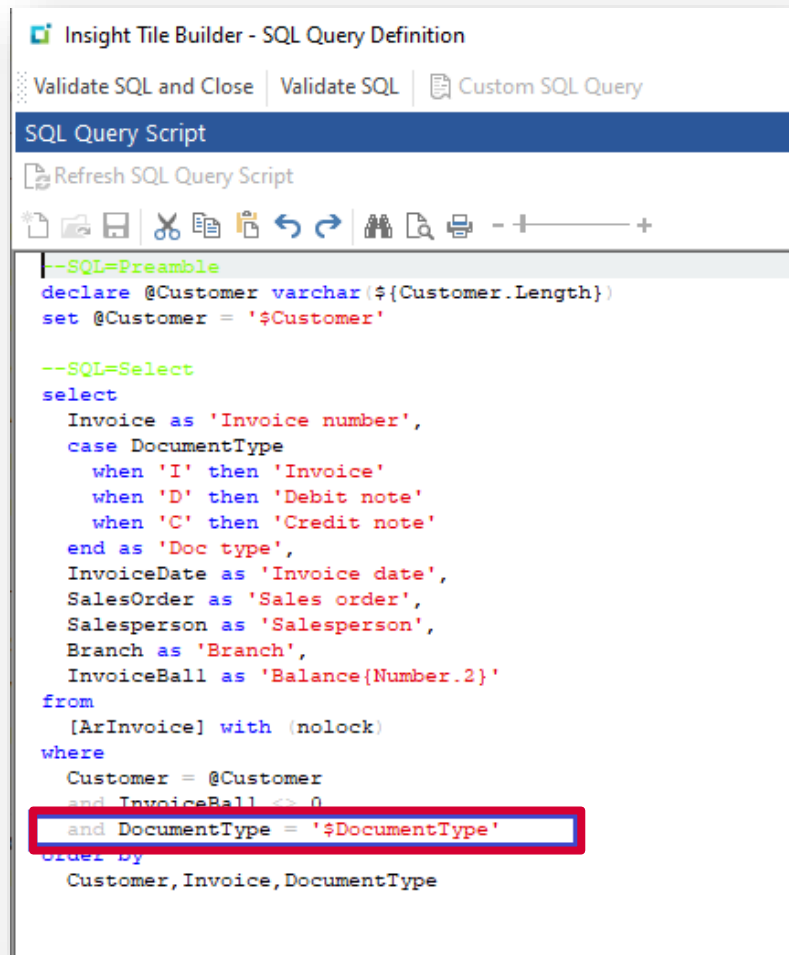
See example below:



You can then reference the variable **\$DocumentType** in your SQL statement as required. The variable **\$DocumentType** will be replaced with the parameter value selected – e.g. 'I'.

For example, we have appended the phrase

and DocumentType = '\$DocumentType'



```
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '$Customer'

--SQL=Select
select
  Invoice as 'Invoice number',
  case DocumentType
    when 'I' then 'Invoice'
    when 'D' then 'Debit note'
    when 'C' then 'Credit note'
  end as 'Doc type',
  InvoiceDate as 'Invoice date',
  SalesOrder as 'Sales order',
  Salesperson as 'Salesperson',
  Branch as 'Branch',
  InvoiceBall as 'Balance{Number.2}'
from
  [ArInvoice] with (nolock)
where
  Customer = @Customer
  and InvoiceBall <> 0
  and DocumentType = '$DocumentType'
order by
  Customer, Invoice, DocumentType
```

There are alternate ways of performing a similar task. Such as the parameter list consisting of the words Invoice, Debit note, or Credit note and using the first character of the parameter **\$DocumentType** or alternatively using a SQL case statement etc. This has just been provided as an example of one technique that could be used.

PARAMETER TYPE – KEYLIST

When defining a parameter type **KeyList**, the system allows you to enter a comma separated list of keys.

There is no validation or other logic performed currently using this parameter type.

PARAMETER USED TO OVERRIDE CONTEXT KEY

Until now we have looked at how you can define a key type against the tile properties and the use the SQL Preamble to use the passed key in context from the application and use that to select specific data to be shown. Earlier in this document we use the Customer code as an example.

In addition, we have seen how you can use a parameter to select a specific key when adding the tile to the user's workspace – In the topic [Parameters - Worked Example](#) we use the example of a branch code.

Suppose you wish to create a single tile that can be used for both purposes. i.e. when adding the tile to a user's workspace you can decide whether to use a hard-coded value (like our Branch example) or use a key value passed from the application context (such as the Customer example).

It is easy to create a tile that handles both.

Simply create a parameter with the same name as the variable passed as a context key.

For example, if you have a tile that uses a context key 'Customer' you can create a tile parameter with the same name 'Customer' (and define it as a Key type 'Customer').

Then when adding the tile to the user's workspace, you will be prompted for the Customer key. If you leave it blank then the context Customer key passed will be used, else the hard-coded Customer code will be used.

You may wish to change exactly how the tile appears in each of these scenarios – see [Alternate Tile Definitions](#).

Alternate tile definitions

If you have a single tile that supports both:

1. A key being passed in context from an application, and,
2. A key to be configured when adding the tile to the user's workspace to override the context key from the application

You may wish to show slightly different information on the Text tile based on the source of the key.

We will use a new example to demonstrate this feature.

The tile will show the total number of sales order lines with a non-zero backorder quantity for a single customer.

If the customer code is determined from the context (for example from the Customer Query or Sales Order Entry applications), then the applications has already shown the customer code and name. Therefore, we will simply show:

Tile part	Value shown
Title	Backorders
SubTitle	
Value	{Count_backorder_lines}
Footer	

However, if we define the customer code when adding the tile to the user's workspace, we need to know the customer code and name – hence the tile will be changed to show:

Tile part	Value shown
Title	Backorders
SubTitle	Customer: {Customer_code}
Value	{Count_backorder_lines}
Footer	{Customer_name}

We have shown 'pseudo' values in curly brackets

The Insight Tile Builder allows you to define the tile parts to represent the first of these styles (i.e. where the context key is derived from the application).

In addition, you can define alternate tile parts that will be used when you define the parameter when adding the tile to the user's workspace. The tile Title is common to both (we will use the phrase 'Backorders' in our example).

You can define alternate tile parts for the SubTitle, Value and Footer components. Once you have defined one of these alternate parts then the primary definitions will be suppressed, and the alternate definitions used instead.

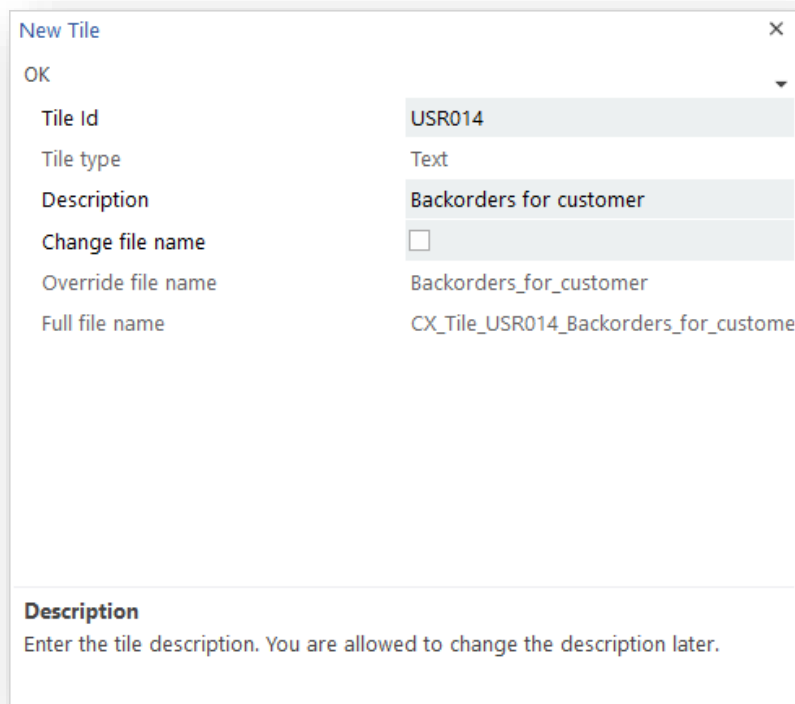
Tile part	Primary values if key passed in context from the application	Alternate value shown if key defined when adding tile to user's workspace
Title	Backorders	{uses primary Title}
SubTitle		Customer: {Customer_code}
Value	{Count_backorder_lines}	{Count_backorder_lines}
Footer		{Customer_name}

The following steps will demonstrate how to add a tile with this capability.

CREATING THE TILE DEFINITION

Load the Insight Tile Definition (IMPKPI) program and select the 'New Tile' toolbar button.

Enter the following at the New Tile dialog.



New Tile [X]

OK [v]

Tile Id: USR014

Tile type: Text

Description: Backorders for customer

Change file name:

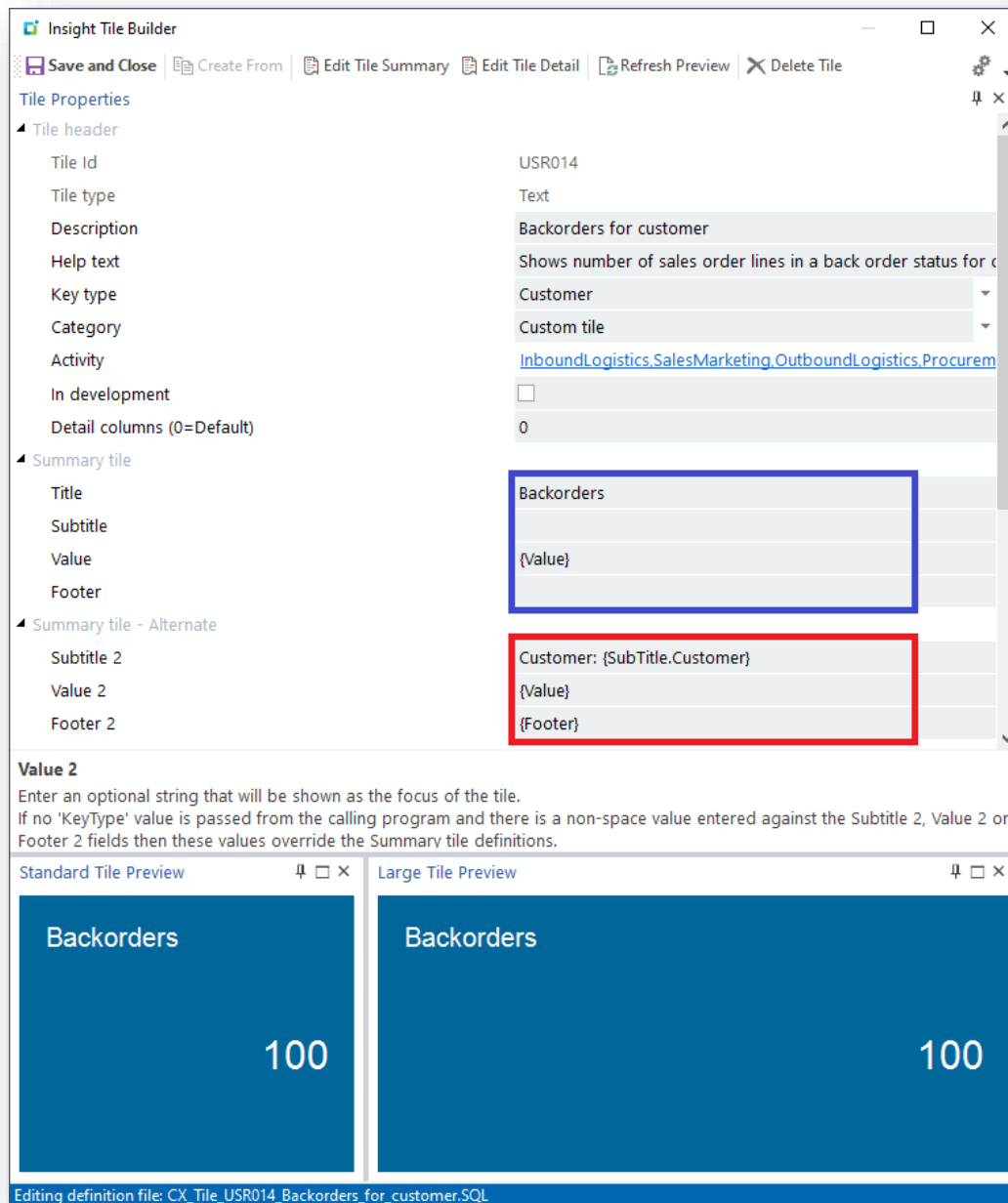
Override file name: Backorders_for_customer

Full file name: CX_Tile_USR014_Backorders_for_custome

Description

Enter the tile description. You are allowed to change the description later.

Update the Insight Tile Builder attributes as shown below:

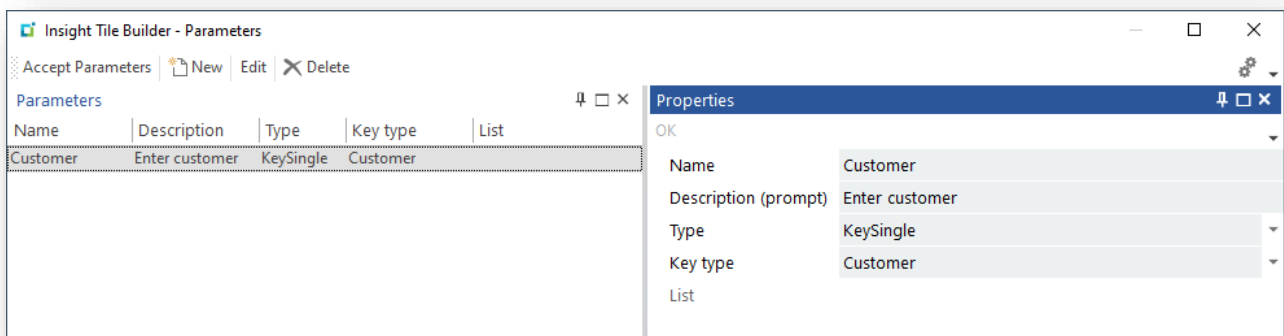


Considerations:

- The key type is defined as 'Customer' – this allows the customer code in context to be passed from the calling application.
- The 'Summary tile' fields are highlighted in blue.
 - These will be used when the customer code is passed in context
- The 'Summary tile – Alternate' fields are highlighted in red.
 - These will be used (together with the Summary Tile Title) when the customer code has been defined at the time the tile is added to the user's workspace.

TILE PARAMETERS

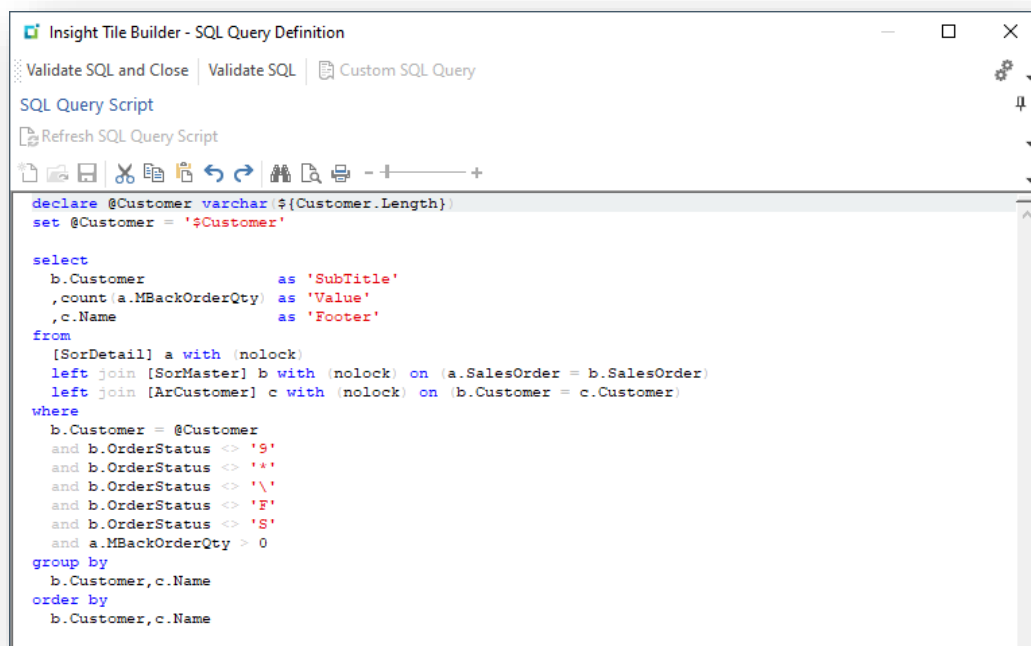
You must then define a single parameter named 'Customer' – as shown below:



This will be used to prompt for the customer code at the time you add the tile to the user's workspace.

SUMMARY TILE SQL

You should then define the tile summary definition and select a 'Custom SQL Query'. The SQL statement shown be the following:



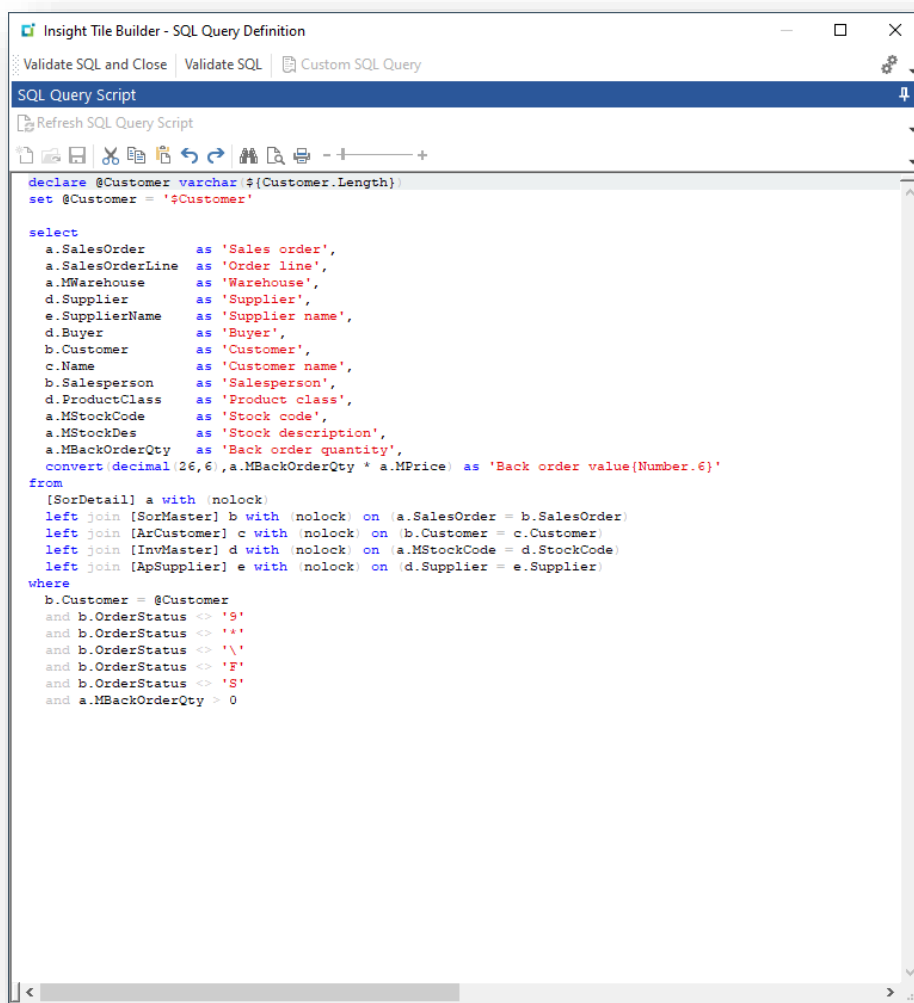
Considerations:

- The column name **SubTitle** returns the customer code
 - This is used by the Alternate tile **{SubTitle}** receiving variable
- The column name **Value** returns an aggregate value – the count of lines
 - This is used by both the primary and alternate tile **{Value}** receiving variable

- The column name **Footer** returns the customer name
 - This is used by the Alternate tile **{Footer}** receiving variable
- The condition excludes an appropriate set of status codes
 - The orders in a completed, cancelled, scheduled, forward order, etc. status
- The condition only includes order lines that have a non-zero back order quantity
- The 'group by' and 'order by' clauses ensure that only a single line is returned for the customer – i.e. we return an aggregate value

DETAIL TILE SQL

You should then define the tile detail definition and select a 'Custom SQL Query'. The SQL statement shown be the following:



```

Insight Tile Builder - SQL Query Definition
Validate SQL and Close | Validate SQL | Custom SQL Query
SQL Query Script
Refresh SQL Query Script
declare @Customer varchar(255) = '{Customer.Length}'
set @Customer = '{Customer}'

select
  a.SalesOrder      as 'Sales order',
  a.SalesOrderLine as 'Order line',
  a.MWarehouse     as 'Warehouse',
  d.Supplier        as 'Supplier',
  e.SupplierName    as 'Supplier name',
  d.Buyer           as 'Buyer',
  b.Customer        as 'Customer',
  c.Name            as 'Customer name',
  b.Salesperson     as 'Salesperson',
  d.ProductClass   as 'Product class',
  a.MStockCode      as 'Stock code',
  a.MStockDes       as 'Stock description',
  a.MBackOrderQty   as 'Back order quantity',
  convert(decimal(26,6), a.MBackOrderQty * a.MPrice) as 'Back order value(Number:6)'
from
  [SorDetail] a with (nolock)
left join [SorMaster] b with (nolock) on (a.SalesOrder = b.SalesOrder)
left join [ArCustomer] c with (nolock) on (b.Customer = c.Customer)
left join [InvMaster] d with (nolock) on (a.MStockCode = d.StockCode)
left join [ApSupplier] e with (nolock) on (d.Supplier = e.Supplier)
where
  b.Customer = @Customer
and b.OrderStatus <> '9'
and b.OrderStatus <> '*'
and b.OrderStatus <> '\ '
and b.OrderStatus <> 'F'
and b.OrderStatus <> 'S'
and a.MBackOrderQty > 0
  
```

Considerations:

- The condition is the same as the summary SQL statement
 - This ensures that the items included in the summary tile are the same ones being shown in the detail drilldown

- We have performed a 'raw' calculation of the backorder value by multiplying the backorder quantity by the price.
 - Depending on the quantity and price unit of measures this may not yield an accurate value
 - The maximum supported decimal value is 20 integers and 6 decimal places. This is a SQL datatype definition of decimal(26,6).
 - In this example, we have used a SQL 'convert' statement to ensure the datatype returned is suitable.
 - See [Appendix 2 – Data Types](#) for more information.
 - We have also used a list edit modifier to ensure that we always show 6 decimal places: 'Back order value{Number.6} '
- Remember to use the **with (nolock)** lock hint for every table referenced.
 - *Failure to do this may result in the statement appearing to hang (block) when any of the rows are involved in an active transaction.*

ADDING A TILE TO USER'S WORKSPACE

We're now going to add this tile twice to the Customer Query application.

When we add the first tile, we're going to leave the customer code as spaces at the 'Enter Customer' prompt. i.e. the customer will be determined from the context by using the current customer code entered in the Customer Query.

When we add the second tile, we're going to hardcode a customer code of '0000001' at the 'Enter customer' prompt. See below:

Once we have saved the edited workspace, returned to the menu, and reloaded the Customer Query application the two tiles will take effect.

VIEWING THE TILES AT RUN TIME

If we load the Customer Query and select customer '0000001' then the tiles will look as follows:



The first instance of the tile is 'in context' so does not show the customer code or name. The second instance shows the hardcoded customer code and its name.

If we select another customer code, then the tile 'in context' shows the new value, whereas the hardcoded tile shows the same value as before:



This example has just been used to demonstrate the capability this may be put to a more practical usage depending on your requirements.

The main purpose of this example is to demonstrate that we can create a single **Insight Tile** that has multiple purposes and avoids us having to create multiple tiles.

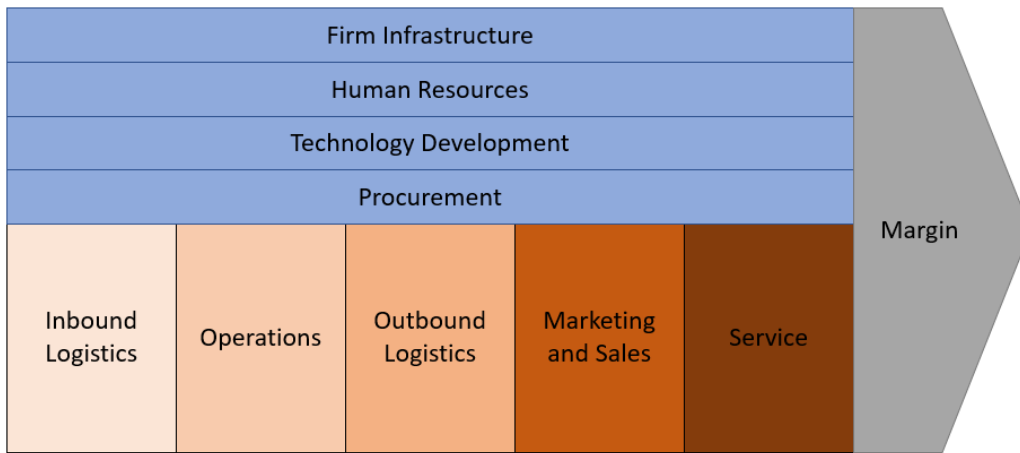
This has additional benefits when taking advantage of the KPI capability – as we only need to define a single set of KPI thresholds as there is only a single tile.

Insight Tile Properties – Business Activities

The Insight Tile Builder allows you to assign one or more business activities to an **Insight Tile** using the activity classification model defined by **Porter’s Value Chain**.

Porter’s Value Chain involves five primary activities: inbound logistics, operations, outbound logistics, marketing and sales, and service. Support activities are illustrated in a vertical column over all the primary activities. These are procurement, human resources, technology development, and firm infrastructure.

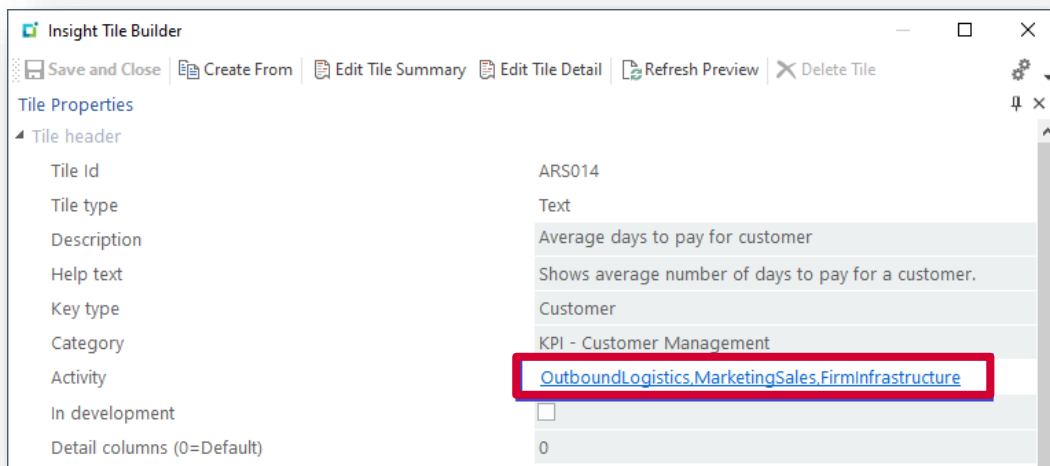
The following diagram represents this model.



You can assign one or more of these activities when configuring an **Insight Tile**.

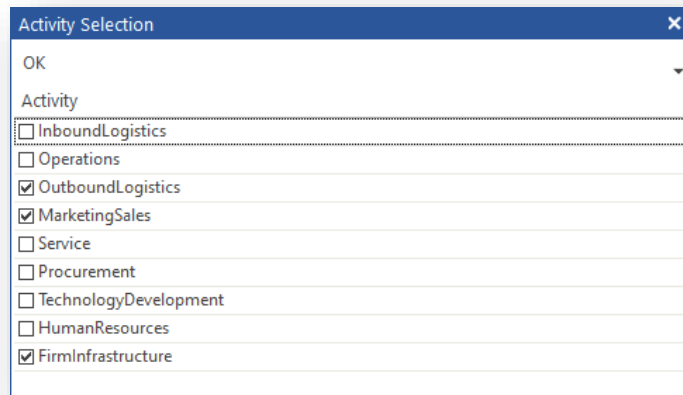
Note: Activities are currently documentary only. However, future releases may support selecting or filtering **Insight Tiles** by Activity and therefore it is good practice to assign appropriate activity selections to your **Insight Tiles**.

For example, the standard tile **Average days to pay for customer** (ARS014) has the following properties:



Note the list of assigned Activities are shown as a comma separated hyperlink.

If you were to copy this tile (for example using the 'Create From' toolbar button), then you can view/edited this list by clicking in the hyperlink – see below:



You can then select or deselect any combination of Activities.

Data Sources and Business Views

When defining an **Insight Tile**, you create a SQL statement to retrieve information to be shown.

The data source can be one of the following:

- SYSPRO base table
- SYSPRO custom form table (ends with '+')
- SYSPRO business view (starts with 'bq_')
- User defined base table
- User defined view

If you are simply selecting columns from a single table then in many cases, you can use the **Generated** method of creating a summary or detail tile definition. This uses a relatively simple point-and-click style form entry, automatically generating the SQL statement that will be used at run time.

When accessing a single table there are a few cases where you need to use the **Custom** method of creating the SQL statement – but in many cases the simpler, **Generated** method, will work.

If you wish to select data from two or more tables and/or perform more sophisticated aggregation, selection criteria, etc., you must use the **Custom** method of creating the SQL statement. In addition, if you have several **Insight Tiles** that are based on the same set of tables, then you will end up with many tiles containing the same or similar relatively complex SQL logic.

One potential solution to this challenge is to create a **Business View** and use that as your data source.

INTRODUCING BUSINESS VIEWS

A **Business View** is a SQL View that has been developed and published using the SYSPRO View Builder application.

This not only allows you to create a SQL View using a simple to use interface (a little like the Insight Tile Builder interface) but also to add the **Business View** definition to the SYSPRO data dictionary. This has the additional benefit of allowing many SYSPRO applications to access your **Business View**, just as if it were a standard SYSPRO base table.

Note: You may see **Business Views** being called **Business Activity Queries** or similar. In all cases we mean the ability to use a SYSPRO application to build a SQL View and publish it to SQL Server and the SYSPRO data dictionary. In the remainder of this document we will mainly use the term **Business View**.

Once you have created and published a **Business View** (which creates a SQL View and updates the SYSPRO data dictionary), you can use this data source in the following SYSPRO technologies:

- Insight Tile Builder
- OData
- SYSPRO Report Writer
- Custom reporting from Crystal Reports
- Generic query business objects that access the database using the data dictionary (COMFCH, COMBRW, COMFND, COMKEY)
- Generic data dictionary query program (DDSBFI)
- Direct SQL access via any other relevant technology

ADVANTAGES OF BUSINESS VIEWS

When you create sophisticated **Insight Tiles** you often require joining multiple tables defining the set of data to be selected. These can be any combination of SYSPRO base tables, custom form tables or user defined tables.

However, this requires specialized knowledge of the SYSPRO database and the logical relationships between tables. As there are more than 1,000 base tables in SYSPRO, this is not a trivial expertise to acquire. Lack of this expertise can limit the ability to create effective **Business Views**.

The solution is for an expert on the SYSPRO SQL table relationships to create a **Business View** and publish it to the SQL database and SYSPRO data dictionary.

This provides the following advantages:

- A user who is relatively unskilled in their knowledge of the SYSPRO database can create an **Insight Tile** using the **Business View** as the data source.
- The complexity of any table joins is hidden.
- The **Business View** can include just the relevant columns that are required. This is a significant benefit as often base tables have many columns that are not relevant or even confusing to an inexperienced user.
- If required, the **Business View** can provide column names that are more suitable for an inexperienced user.
- Other technologies (not just **Insight Tiles**) can also use the same **Business View** – again removing the requirement for more expert knowledge and preventing accidental differences between technologies where you require the same data source logic to be applied.

This document will not explain all the **Business View** use cases, or how to use the SYSPRO Business View Builder, however we will provide a simple example to demonstrate some of the values mentioned.

CREATING A BUSINESS VIEW – WORKED EXAMPLE

Suppose we wish to create an **Insight Tile** that queries active sales orders by customer.

The primary table will be the Customer master table **[ArCustomer]** and we will link to the parallel table that stores the customer balances: **[ArCustomerBal]**

We will also create a link to the Customer custom form table **[ArCustomer+]** to include custom information outside of the SYSPRO standard tables.

There will also be a link to the Sales Order master table: **[SorMaster]**

For your information: the common link between all these tables is a column named **Customer**.

We have assumed two columns in the customer custom form table named: **Forecast** and **Region**

BUSINESS ACTIVITY QUERY BUILDER

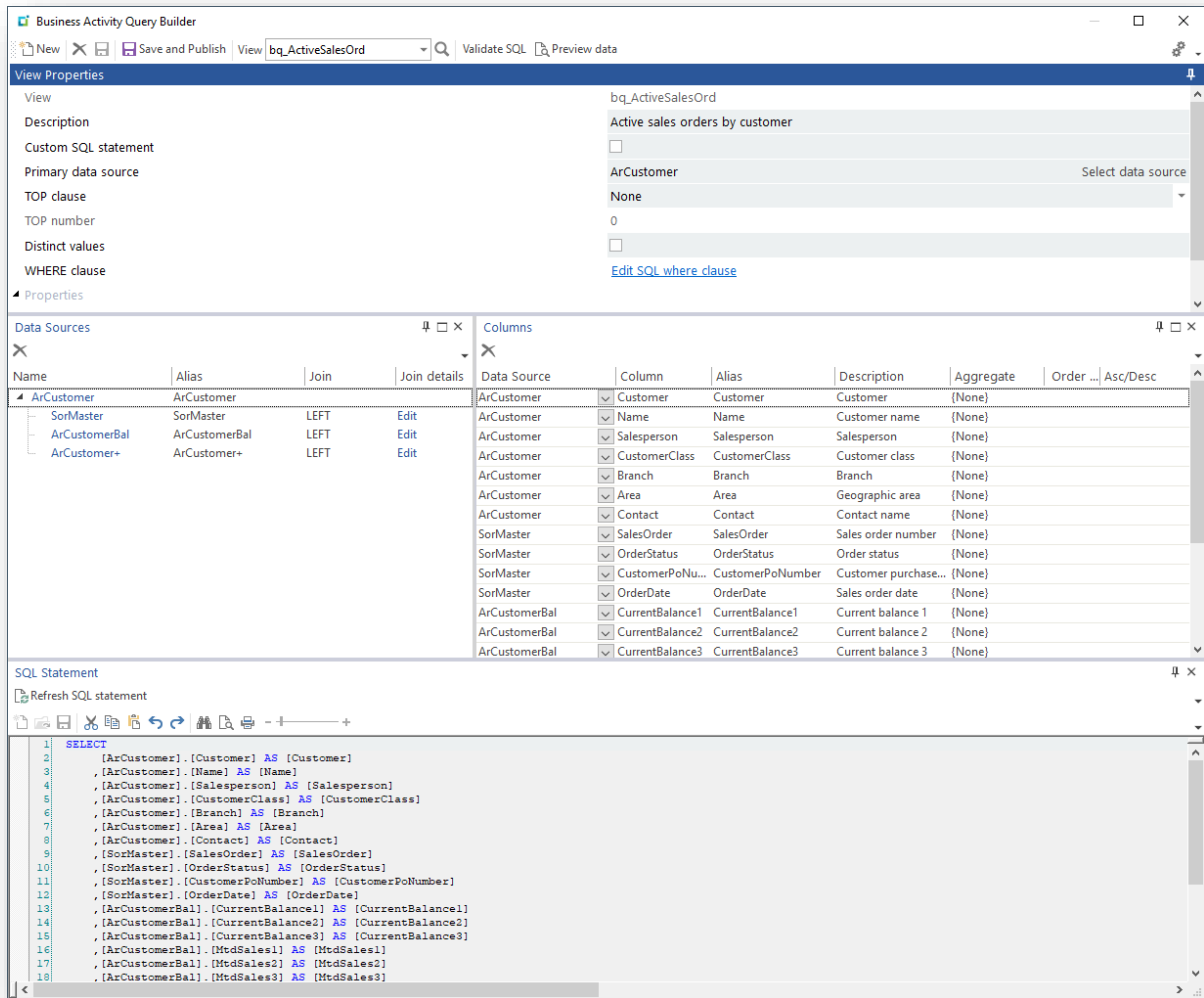
Run the Business Activity Query Builder (IMPVWB) and create a **Business View** named **bq_ActiveSalesOrd**.

The primary data source is **ArCustomer**.

Then we will use the Data Sources pane to add the tables with their relationships.

Then we will select the columns that we wish to include in the **Business View**. This can include columns from any of the source tables. We can also aggregate data and provide computed columns if required.

See the Business Activity Query Builder properties image as follows:



As you are selecting tables and columns, the SQL statement that will be used as part of the SQL View creation being generated, is shown at the bottom of the screen.

Note: Like the Insight Tile Builder, you can use this generated SQL statement as the basis for a completely custom user defined SQL statement – select the checkbox ‘Custom SQL statement’. In our example we will use the generated SQL statement as it suits our purpose.

The complete statement generated is:

```
SELECT
  [ArCustomer].[Customer] AS [Customer]
  , [ArCustomer].[Name] AS [Name]
  , [ArCustomer].[Salesperson] AS [Salesperson]
  , [ArCustomer].[CustomerClass] AS [CustomerClass]
  , [ArCustomer].[Branch] AS [Branch]
  , [ArCustomer].[Area] AS [Area]
  , [ArCustomer].[Contact] AS [Contact]
  , [SorMaster].[SalesOrder] AS [SalesOrder]
  , [SorMaster].[OrderStatus] AS [OrderStatus]
  , [SorMaster].[CustomerPoNumber] AS [CustomerPoNumber]
  , [SorMaster].[OrderDate] AS [OrderDate]
  , [ArCustomerBal].[CurrentBalance1] AS [CurrentBalance1]
  , [ArCustomerBal].[CurrentBalance2] AS [CurrentBalance2]
  , [ArCustomerBal].[CurrentBalance3] AS [CurrentBalance3]
  , [ArCustomerBal].[MtdSales1] AS [MtdSales1]
  , [ArCustomerBal].[MtdSales2] AS [MtdSales2]
  , [ArCustomerBal].[MtdSales3] AS [MtdSales3]
  , [ArCustomerBal].[MtdProfit1] AS [MtdProfit1]
  , [ArCustomerBal].[MtdProfit2] AS [MtdProfit2]
  , [ArCustomerBal].[MtdProfit3] AS [MtdProfit3]
  , [ArCustomer+].[Forecast] AS [Forecast]
  , [ArCustomer+].[Region] AS [Region]
FROM [ArCustomer] AS [ArCustomer]
LEFT JOIN [SorMaster] AS [SorMaster] ON (([ArCustomer].[Customer] = [SorMaster].[Customer] AND
[SorMaster].[ActiveFlag] = ''))
LEFT JOIN [ArCustomerBal] AS [ArCustomerBal] ON (([ArCustomer].[Customer] =
[ArCustomerBal].[Customer]))
LEFT JOIN [ArCustomer+] AS [ArCustomer+] ON (([ArCustomer].[Customer] = [ArCustomer+].[Customer]))
```

During the development of the SQL statement you can click the 'Validate SQL' function to ensure that the statement passes the basic SQL validation tests to ensure that the statement is suitable for a SQL View.

Click 'Preview data' to view the data that would be returned.

Warning: When previewing data, the system may attempt to select an excessive number of rows and slow your system.

Once you are happy with the **Business View**, select 'Save and Publish' to have the SQL View physically inserted into your database and the information about the view added to the SYSPRO data dictionary. You will be prompted to overwrite any previous SQL View with the same name.

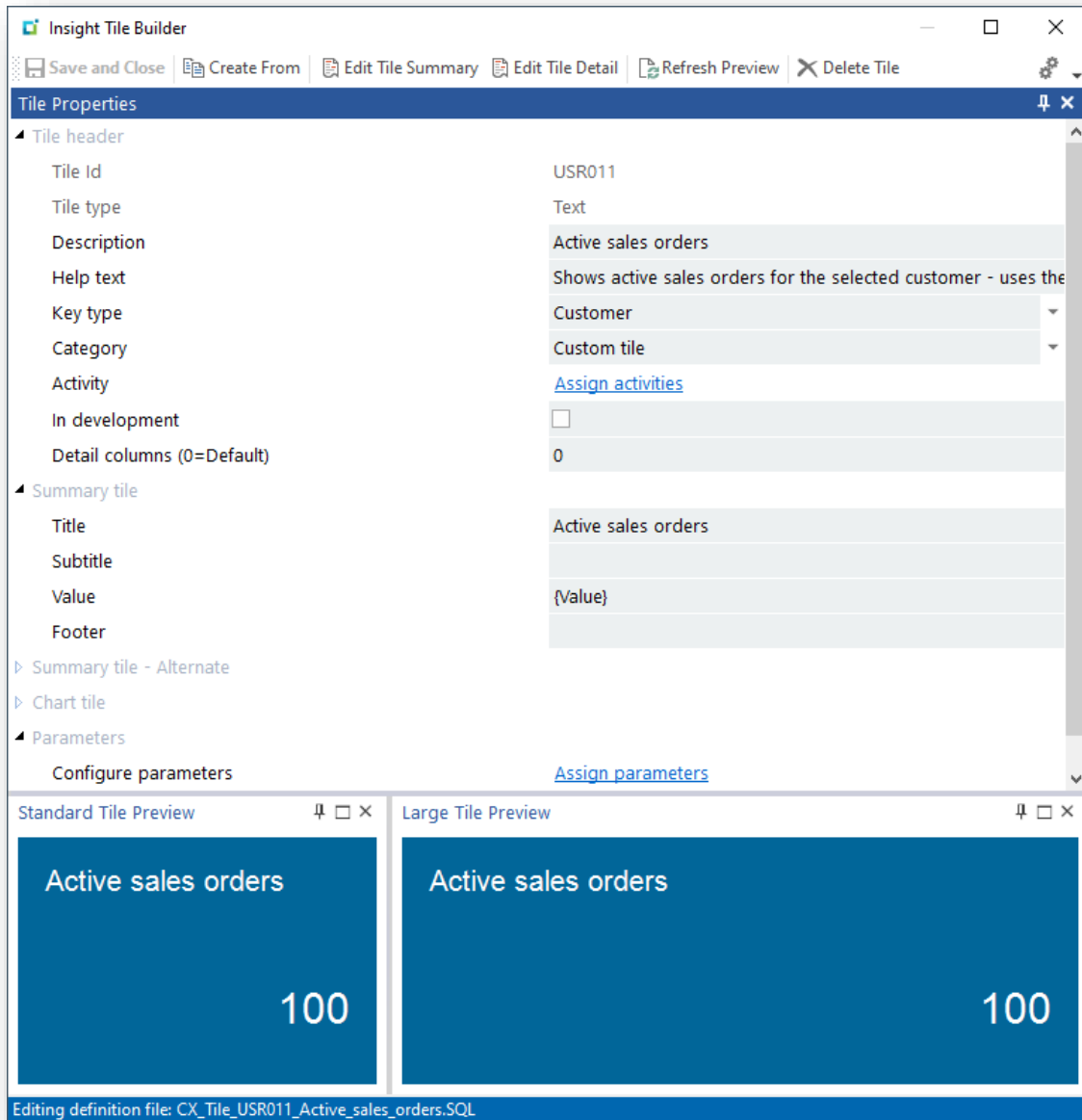
If you have multiple company databases, you will be prompted to select which company databases you wish the SQL View to be applied. Select the companies relevant to your situation.

You can return to the view builder application and change the view at any time – when you have made any changes you should always select the 'Save and Publish' for any changes to take effect.

Once the **Business View** has been created and published, we are ready to use this as a data source in the Insight Tile Builder.

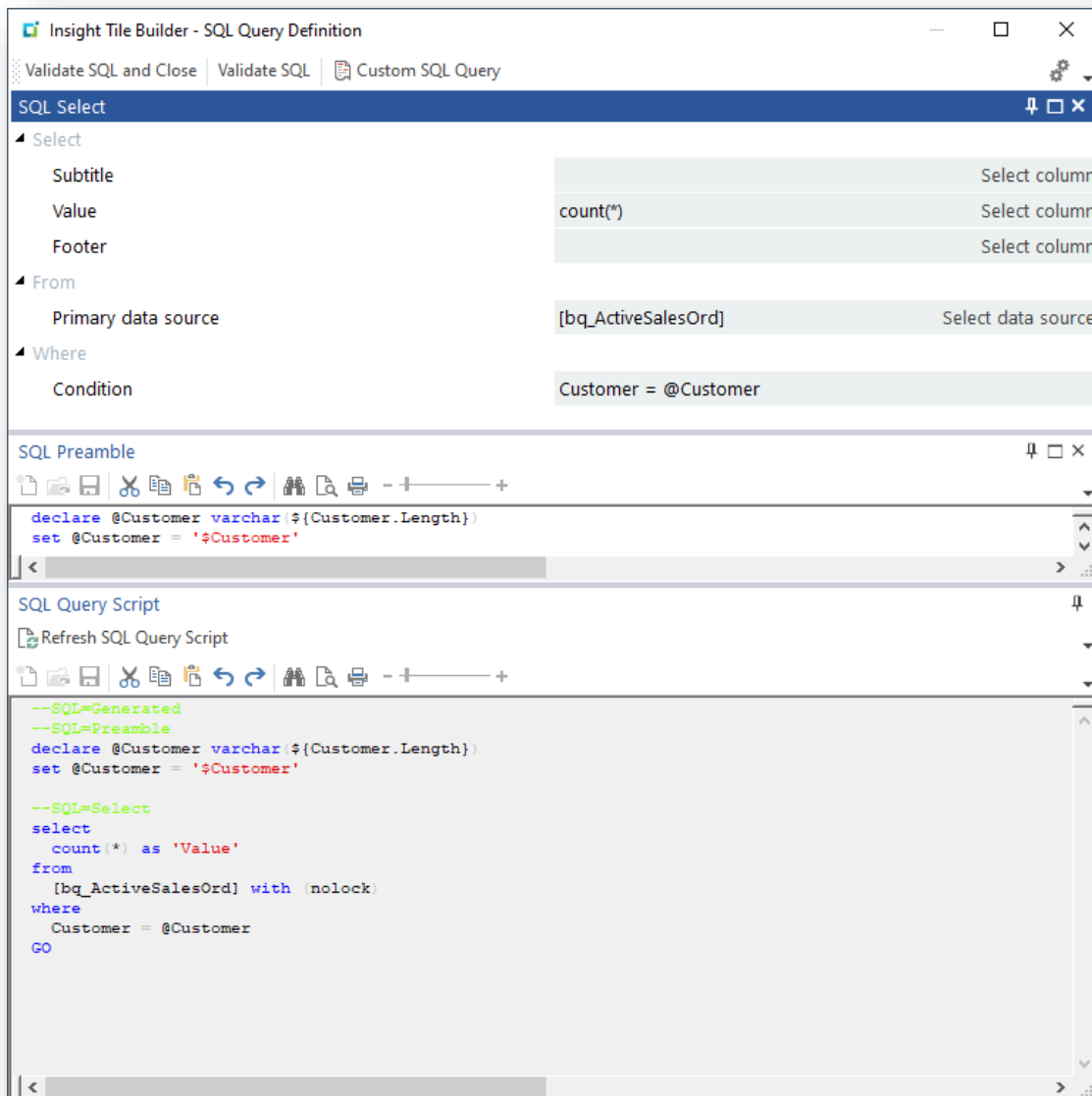
CREATING AN INSIGHT TILE USING A BUSINESS VIEW

Use the Insight Tile Builder and add a text tile with the following properties:



Note the Key type must be 'Customer'.

Then select the 'Edit Tile Summary' toolbar function and define the SQL Query Definition:

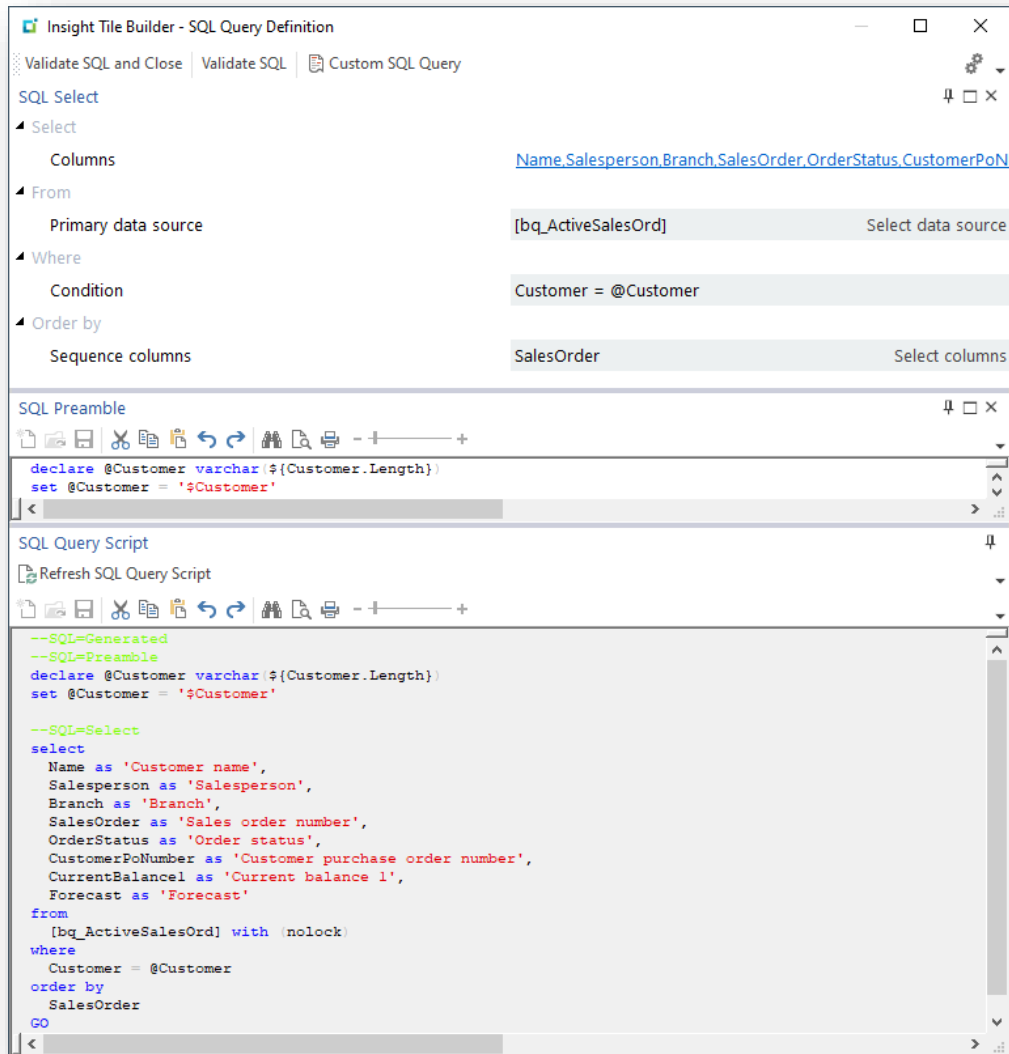


Note that the Primary data source is the **Business View** named **bq_ActiveSalesOrd**

In this case we just want a count of the number of rows for the selected customer, so the Value field is defined as: **count(*)**

You will notice that the generated SQL definition is extremely simple as we are using the power of the **Business View** to remove any complexity. The user may not even realize which base tables are being used.

Next, we are going to use the 'Edit Tile Detail' toolbar function from the main Insight Tile Builder application to create the detail drilldown definition.



In this case we used the Primary data source **bq_ActiveSalesOrd**, the same as the summary definition, and selected various columns.

Again, notice how simple the generated SQL definition is considering that we have selected data from the following underlying tables:

Column name	Source table	Comment
Name	ArCustomer	
Salesperson	ArCustomer	
Branch	ArCustomer	
SalesOrder	SorMaster	
OrderStatus	SorMaster	
CustomerPoNumber	SorMaster	
CurrentBalance1	ArCustomerBal	
Forecast	ArCustomer+	Custom form field

Once saved, add the new tile to the user's workspace in the Customer Query application.

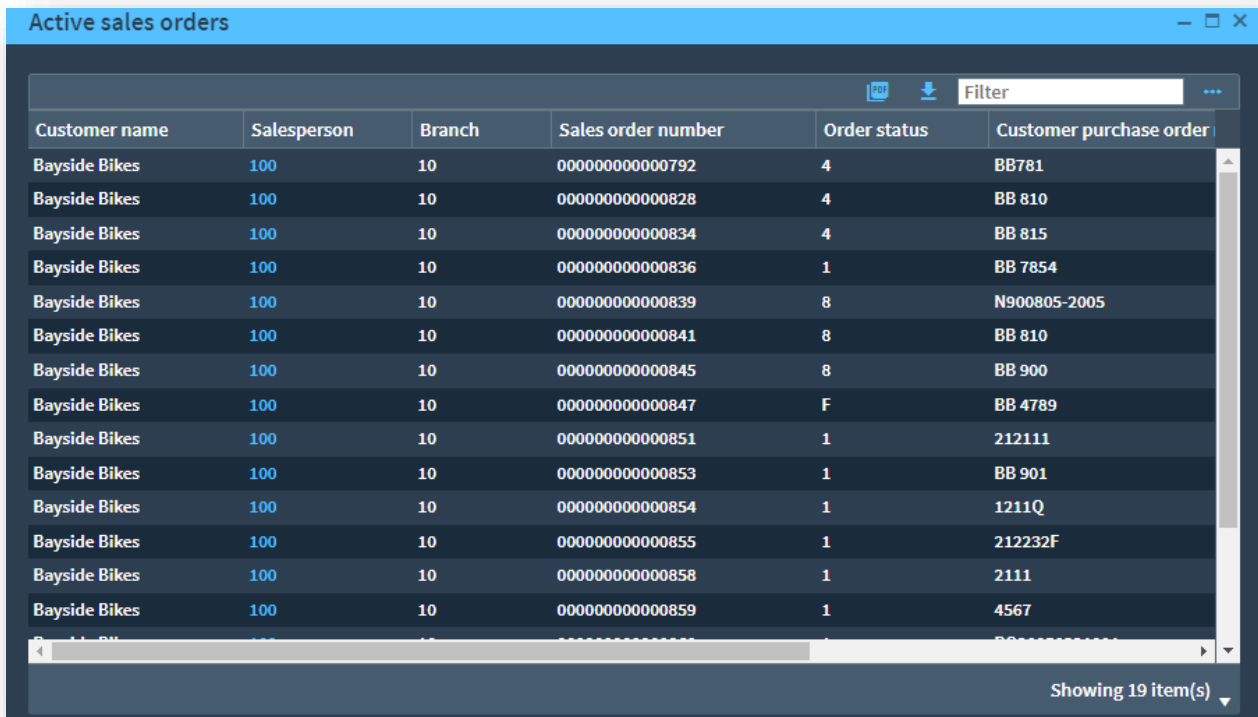
Remember that we have assigned this tile to a category named 'custom tile' to make it easy to find amongst the potentially long list of available tiles.

When you select a customer in the Customer Query the tile will be refreshed – for example:



In this case there are 19 active sales orders.

When you click on the tile, the detail drilldown will show as follows:



Customer name	Salesperson	Branch	Sales order number	Order status	Customer purchase order
Bayside Bikes	100	10	000000000000792	4	BB781
Bayside Bikes	100	10	000000000000828	4	BB 810
Bayside Bikes	100	10	000000000000834	4	BB 815
Bayside Bikes	100	10	000000000000836	1	BB 7854
Bayside Bikes	100	10	000000000000839	8	N900805-2005
Bayside Bikes	100	10	000000000000841	8	BB 810
Bayside Bikes	100	10	000000000000845	8	BB 900
Bayside Bikes	100	10	000000000000847	F	BB 4789
Bayside Bikes	100	10	000000000000851	1	212111
Bayside Bikes	100	10	000000000000853	1	BB 901
Bayside Bikes	100	10	000000000000854	1	1211Q
Bayside Bikes	100	10	000000000000855	1	212232F
Bayside Bikes	100	10	000000000000858	1	2111
Bayside Bikes	100	10	000000000000859	1	4567

Showing 19 item(s)

Note: At run time, when we click on the tile, we will only be showing information relating to a single customer so the columns relating to the customer will be duplicated in the drilldown – in this case columns from **ArCustomer**, **ArCustomerBal** and **ArCustomer+**. Notice that the columns from the sales order table (**SorMaster**) vary from line to line.

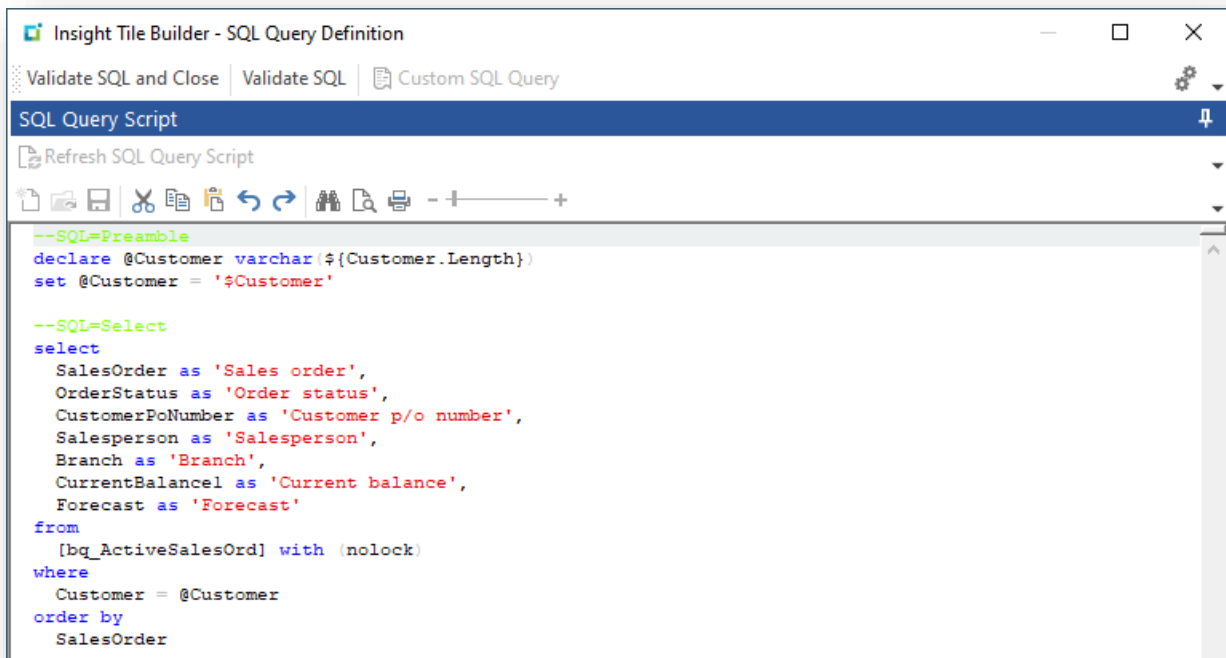
MINOR CUSTOMIZATION TO IMPROVE THE DETAIL DRILLDOWN

As explained previously, once you have initially created and viewed the tile, you may want to customize the tile definition to further improve the user experience.

In this case we will change the column description for the sales order column to 'Sales order' as the drilldown list logic better understands that this is a sales order and will apply key formatting preferences – it will also provide a hyperlink to be able to drilldown into the sales order details.

We will also remove the Customer name from the list as this is visible from the Customer Query – you could remove the other Customer master fields such as Salesperson and Branch (but in our example we will retain them).

We have also changed the order in which the columns will be presented and abbreviated some of the descriptions. The tile detail custom SQL definition looks as follows:



```
--SQL=Preamble
declare @Customer varchar({Customer.Length})
set @Customer = '$Customer'

--SQL=Select
select
  SalesOrder as 'Sales order',
  OrderStatus as 'Order status',
  CustomerPoNumber as 'Customer p/o number',
  Salesperson as 'Salesperson',
  Branch as 'Branch',
  CurrentBalance as 'Current balance',
  Forecast as 'Forecast'
from
  [bq_ActiveSalesOrd] with (nolock)
where
  Customer = @Customer
order by
  SalesOrder
```

At run time the detail drilldown looks as follows:

Sales order	Order status	Customer p/o number	Salesperson	Branch	Current balance	Forecast
000792	4	BB781	100	10	1,969,183.05	100,000
000828	4	BB 810	100	10	1,969,183.05	100,000
000834	4	BB 815	100	10	1,969,183.05	100,000
000836	1	BB 7854	100	10	1,969,183.05	100,000
000839	8	N900805-2005	100	10	1,969,183.05	100,000
000841	8	BB 810	100	10	1,969,183.05	100,000
000845	8	BB 900	100	10	1,969,183.05	100,000
000847	F	BB 4789	100	10	1,969,183.05	100,000
000851	1	212111	100	10	1,969,183.05	100,000
000853	1	BB 901	100	10	1,969,183.05	100,000
000854	1	1211Q	100	10	1,969,183.05	100,000
000855	1	212232F	100	10	1,969,183.05	100,000
000858	1	2111	100	10	1,969,183.05	100,000
000859	1	4567	100	10	1,969,183.05	100,000
000860	1	PO20050321001	100	10	1,969,183.05	100,000

BENEFITS OF INSIGHT TILES USING BUSINESS VIEW DATA SOURCES

In summary, by using a **Business View** we have been able to hide the underlying complexity from the user creating an **Insight Tile**.

This has allowed us to take advantage of the simple point-and-click generation of the Insight Tile SQL statements - avoiding having to delve into creating SQL customized statements that often require more detailed SQL expertise.

Even when we have customized a SQL statement (in the example we customized the detail drilldown), the customization was simple – just re-ordering the columns in the select statement and rewording the column descriptions.

This not only simplifies creating of any **Insight Tiles** based on this **Business View**, but also, we can use the same **Business View** for other query or reporting technologies, and we can be sure that the same data relationships are being used in all cases.

Lastly, we have demonstrated how to incorporate custom data, in this case custom form fields, allowing you to extend the data source reach almost indefinitely. It would be just as easy to include data from any user defined tables.

Charts

Until now the majority of the Insight Tile Reference Guide has focussed on text tiles.

Setting up one or more text tiles on a user's workspace can provide relevant insight, guiding behaviour and adding value to the business.

However, the Insight Tile system also allows charts to be shown. These can provide trends and comparisons in a way that a simple text tile never could.

The remainder of this topic will introduce the different chart types and how to create them using the Insight Tile Builder by using some worked examples.

Just as an example, the following chart tile represents the number of users logged into SYSPRO over the past few weeks compared to the number of licensed users.



As you can see, it provides an at-a-glance indication that sometimes our site gets close to the number of licensed users.

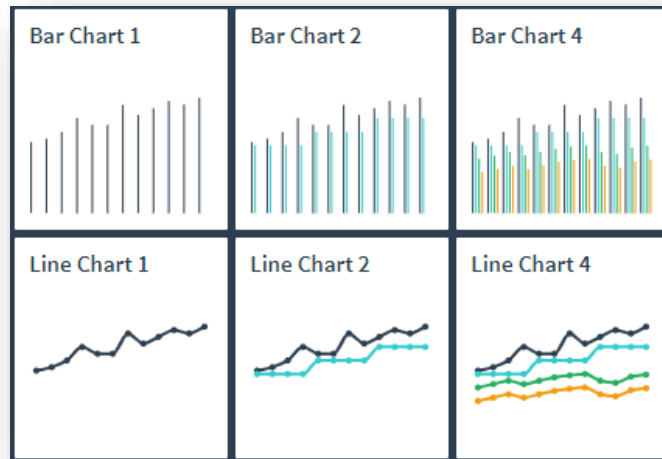
Note how simple the chart is – the user can move the mouse over the points to see the data points. Even though the chart is clean and simple the trends are clear to see.

CHART TYPES: BAR AND LINE

There are currently two supported chart types – bar charts and line charts.

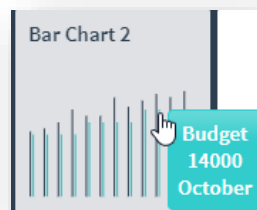
They both allow you to define up to 4 series and each series can have up to 24 data points.

The following example set of charts was created to demonstrate the different types and a sample showing 1, 2 and 4-series.

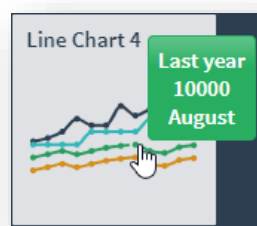


The tooltip values that are shown as you mouse-over a chart allows the user to view the data points. The contents of the tooltip are configurable when designing the tile.

The following shows the tooltip over the 2-series bar chart – in this case one of the ‘Budget’ series data points.



The following shows the tooltip over the 4-series line chart – in this case one of the ‘Last year’ series data points.



CREATING A CHART TYPE INSIGHT TILE

The reason that chart type **Insight Tiles** have been left to later in the Insight Tile Reference guide is that although there are some similarities to text tiles, chart tiles are also quite different.

In addition, you will require more specialized SQL knowledge and experience when creating chart tiles compared to most text tiles.

When creating a chart tile, you must first decide how many series you require – the system supports 1, 2, 3 or 4 series.

You will be creating a SQL statement that returns the same number of rows as there are series. The first row returned will contain the values for the first series and so on. If you only have a single series, then you must only return a single row.

Each row must contain the data points – the data point values are determined from SQL column names 'P1' thru 'P24'. If you require a chart with (say) 12 data points, then you will be returning values for SQL columns named 'P1' thru 'P12'.

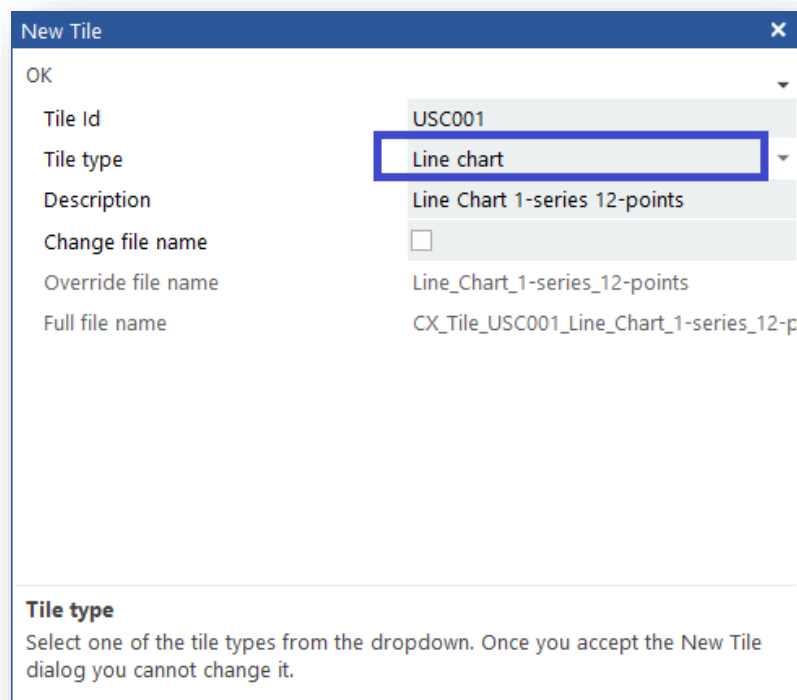
In addition, you can define a tooltip that represents the X-Axis label.

SINGLE SERIES CHART DEMONSTRATION

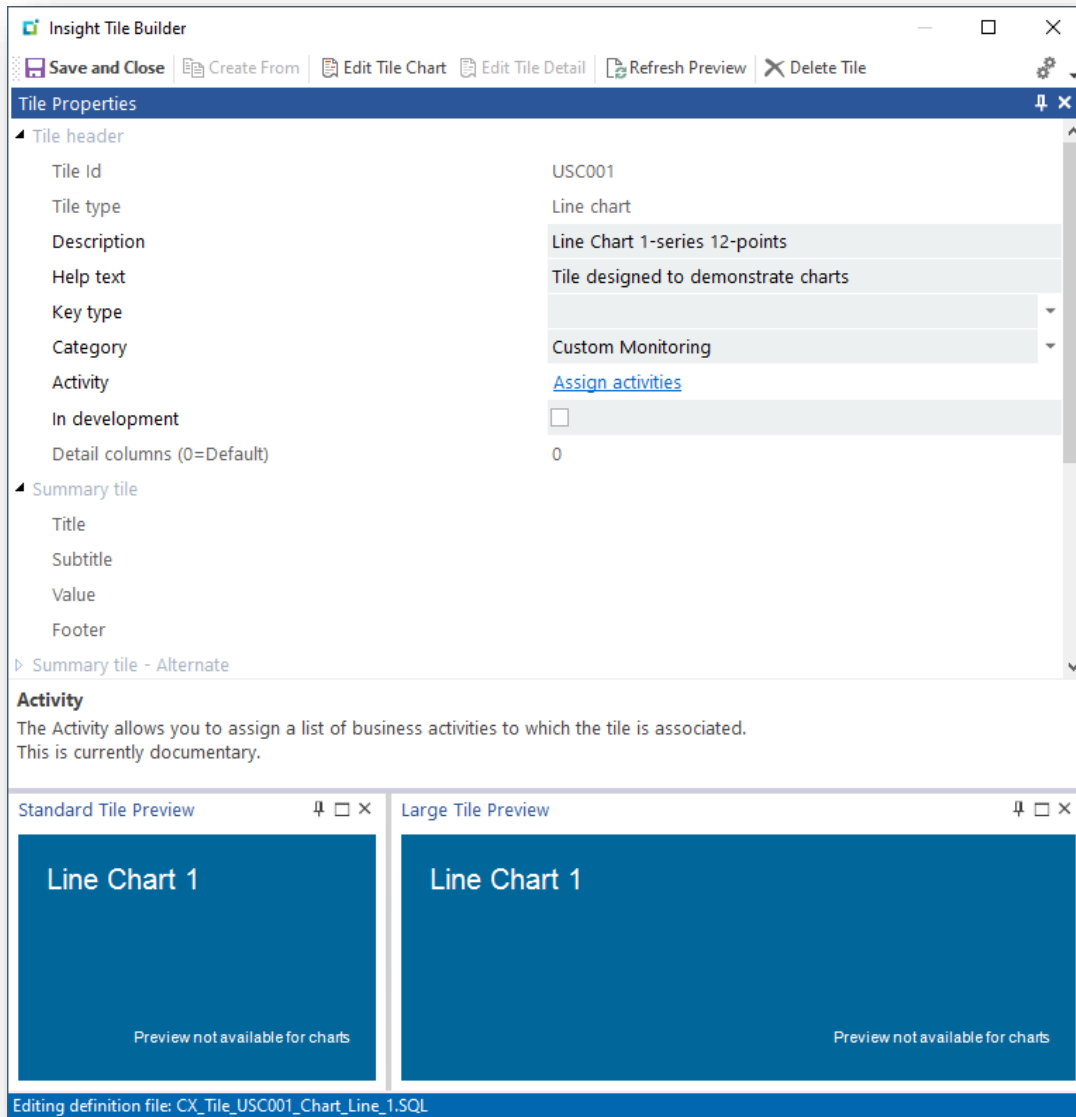
In the following example we are not actually going to query SQL data, but we are going to create a SQL statement that returns hard coded data points for a single series chart. This will allow us to introduce the chart concepts – we will use a real example later in the chapter.

We are going to start with a single series line chart.

Use the Insight Tile Builder application to add a new tile – remember to select 'Line chart' from the Tile type dropdown in the New Tile dialog.

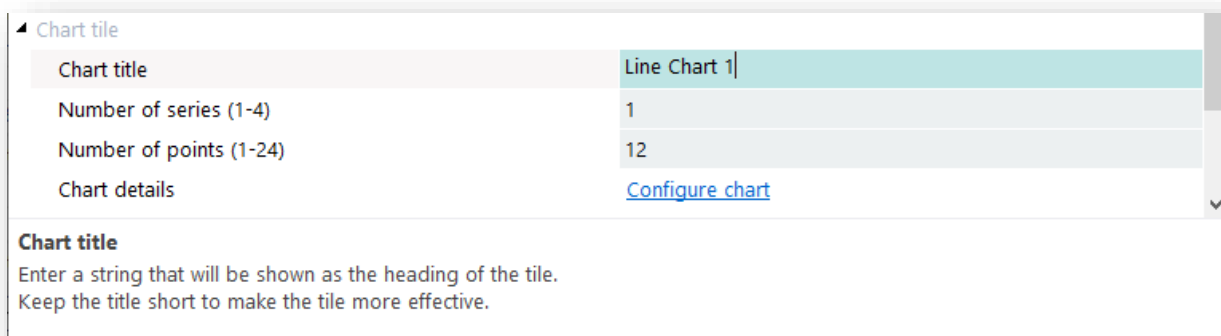


The Tile properties dialog will then be shown:



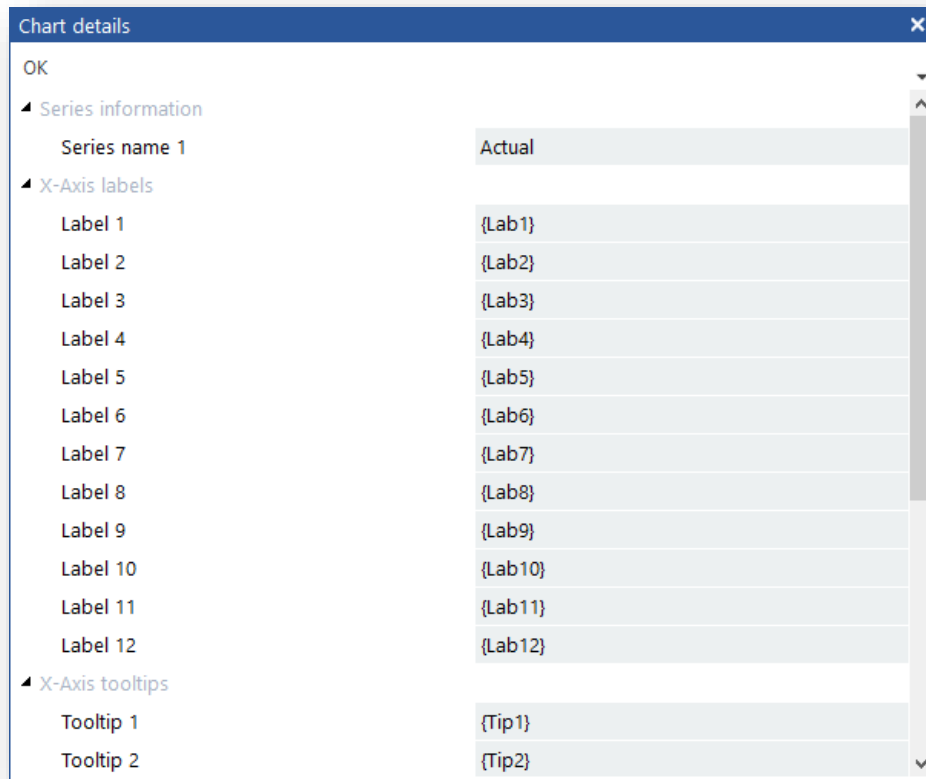
The tile previews are not available when building chart tiles.

Scroll down and expand the 'Chart tile' group:



We have configured the Chart title as 'Line Chart 1', the number of series as 1 and the number of points as 12.

Then click on the 'Configure chart' hyperlink. The following dialog is shown:



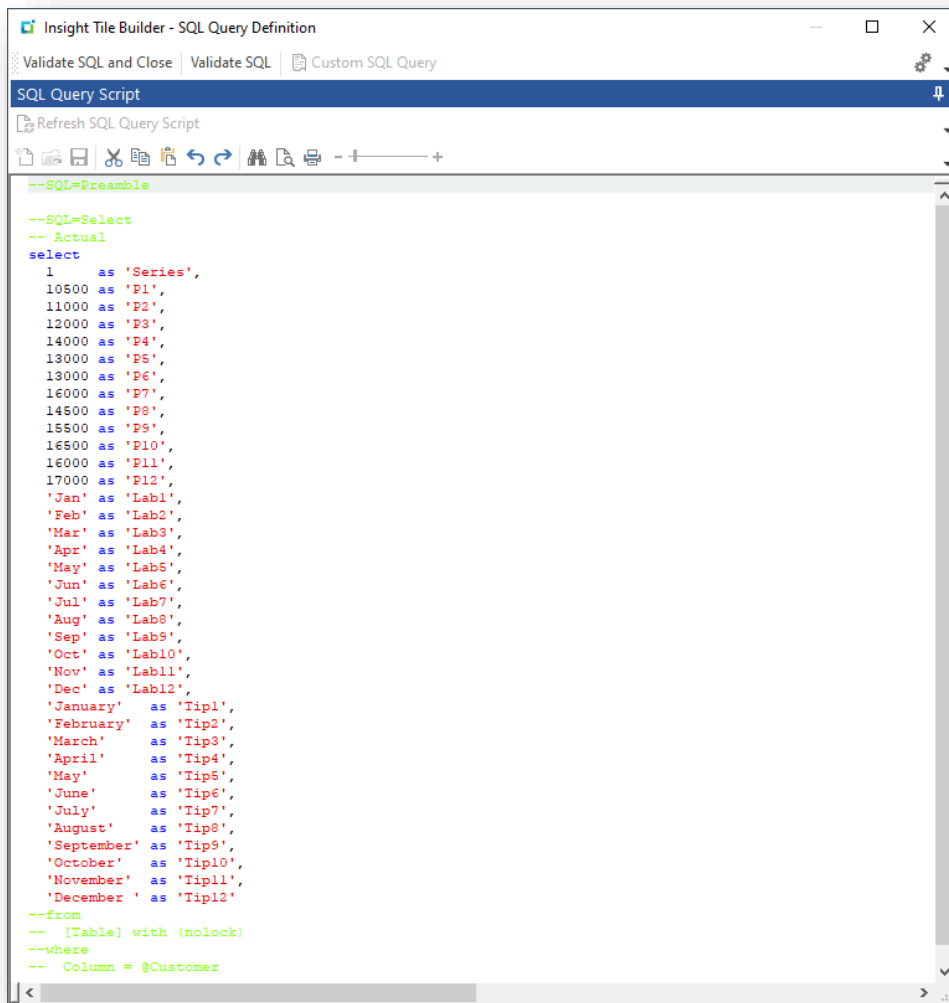
Enter the series name 'Actual' and retain the X-Axis labels and X-Axis tooltips with the predetermined values.

These represent **receiving variables** and are populated at run time by returning SQL columns that exactly match the variable name. For example, receiving variable **{Tip1}** will be populated with a column value from a column named 'Tip1'.

The series name will be visible at run time when you mouse-over data points from the series. The X-Axis labels are documentary and the X-Axis tooltip allows you to append a tooltip value based on the X-Axis position. **You must still define unique X-Axis labels for each value else the data point will not be shown on the chart.**

Click 'OK' to accept these values and return to the main Insight Tile Builder dialog. Then click on the "Edit Tile Chart" toolbar button to load the SQL Query Definition dialog. Chart type tiles always require a **Custom** SQL statement to be defined.

In our example we will be using the following SQL statement:



The screenshot shows a window titled "Insight Tile Builder - SQL Query Definition". The window has a menu bar with "Validate SQL and Close", "Validate SQL", and "Custom SQL Query". Below the menu bar is a tab labeled "SQL Query Script" with a "Refresh SQL Query Script" button. A toolbar contains icons for file operations and editing. The main area displays a SQL query script with the following content:

```
--SQL=Preamble

--SQL=Select
-- Actual
select
  1      as 'Series',
  10500 as 'P1',
  11000 as 'P2',
  12000 as 'P3',
  14000 as 'P4',
  13000 as 'P5',
  13000 as 'P6',
  16000 as 'P7',
  14500 as 'P8',
  15500 as 'P9',
  16500 as 'P10',
  16000 as 'P11',
  17000 as 'P12',
  'Jan' as 'Lab1',
  'Feb' as 'Lab2',
  'Mar' as 'Lab3',
  'Apr' as 'Lab4',
  'May' as 'Lab5',
  'Jun' as 'Lab6',
  'Jul' as 'Lab7',
  'Aug' as 'Lab8',
  'Sep' as 'Lab9',
  'Oct' as 'Lab10',
  'Nov' as 'Lab11',
  'Dec' as 'Lab12',
  'January' as 'Tip1',
  'February' as 'Tip2',
  'March' as 'Tip3',
  'April' as 'Tip4',
  'May' as 'Tip5',
  'June' as 'Tip6',
  'July' as 'Tip7',
  'August' as 'Tip8',
  'September' as 'Tip9',
  'October' as 'Tip10',
  'November' as 'Tip11',
  'December' as 'Tip12'
--from
-- [Table] with (nolock)
--where
-- Column = @Customer
```

For clarity this is shown as text below:

```
-- Actual
select
  1      as 'Series',
  10500 as 'P1',
  11000 as 'P2',
  12000 as 'P3',
  14000 as 'P4',
  13000 as 'P5',
  13000 as 'P6',
  16000 as 'P7',
  14500 as 'P8',
  15500 as 'P9',
  16500 as 'P10',
  16000 as 'P11',
  17000 as 'P12',
  'Jan' as 'Lab1',
  'Feb' as 'Lab2',
  'Mar' as 'Lab3',
  'Apr' as 'Lab4',
  'May' as 'Lab5',
  'Jun' as 'Lab6',
  'Jul' as 'Lab7',
  'Aug' as 'Lab8',
  'Sep' as 'Lab9',
  'Oct' as 'Lab10',
  'Nov' as 'Lab11',
```

```
'Dec' as 'Lab12',
'January' as 'Tip1',
'February' as 'Tip2',
'March' as 'Tip3',
'April' as 'Tip4',
'May' as 'Tip5',
'June' as 'Tip6',
'July' as 'Tip7',
'August' as 'Tip8',
'September' as 'Tip9',
'October' as 'Tip10',
'November' as 'Tip11',
'December' as 'Tip12'
```

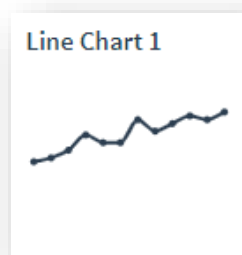
Column notes

- Column 'Series'
 - This column is not required in this example, but later we will expand the examples to show more series, we will be sorting the rows on this Column, so '1' represents that the values returned are for series 1.
- Columns 'P1' thru 'P12'
 - As our sample has 12 data points, we must return columns named 'P1' thru 'P12'.
 - Charts support up to 24 data points.
- Columns 'Lab1' thru Lab12'
 - These are X-axis labels
 - You must include a unique X-Axis label for each point to have the data point shown on the chart
 - X-axis labels are documentary for this version of the Insight Tile logic – but you must still include a unique value for each point to have the data point shown
- Columns 'Tip1' thru 'Tip12'
 - The values returned by these SQL columns are appended to the tooltip along with the series name and data point value.

Note: This is just a demonstration to show the basics of creating a chart. The data point values have been hard coded in the SQL select statement (returned as 'P1' thru 'P12'). Similarly, the X-Axis tooltip values are also hard coded as the months of the year (returned as 'Tip1' thru 'Tip12')

Click 'Validate and Close' and then 'Save and Close' to complete this tile definition.

You can now add the tile to the user's workspace. At run time the tile will look as follows:



Considerations:

- The tile title 'Line Chart 1' is shown at the top of the tile
- The data points 'P1' thru 'P12' are shown as a line chart

You can mouse-over the data points to see more information



Considerations:

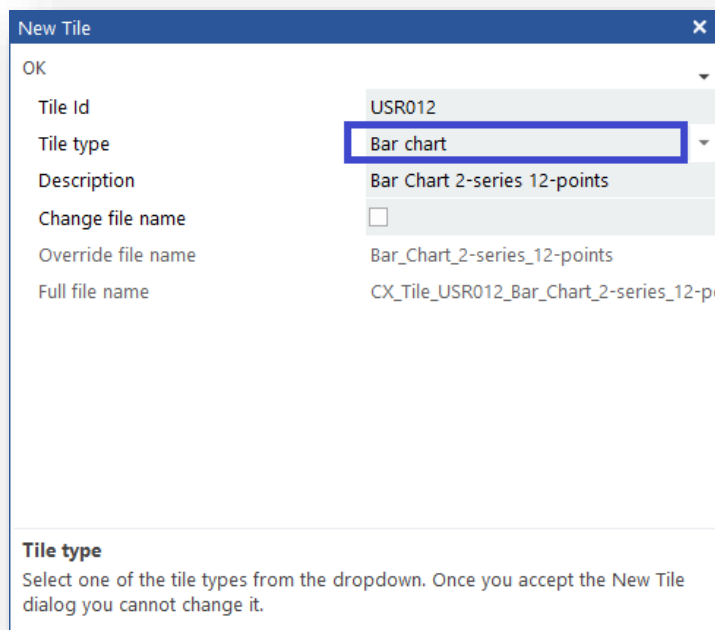
- The phrase 'Actual' is the series name from the 'configure chart' dialog
- The data value 13000 is the value returned from SQL column 'P6'
- The phrase 'June' is the X-Axis tooltip return from SQL column 'Tip6'

From this relatively simple (and completely hard coded) example, we can expand this to handle (say) two series.

TWO SERIES CHART DEMONSTRATION

In this example we will create a bar chart with 2-series.

Add a new tile and select Tile type: Bar chart:



OK	
Tile Id	USR012
Tile type	Bar chart
Description	Bar Chart 2-series 12-points
Change file name	<input type="checkbox"/>
Override file name	Bar_Chart_2-series_12-points
Full file name	CX_Tile_USR012_Bar_Chart_2-series_12-p

Tile type
Select one of the tile types from the dropdown. Once you accept the New Tile dialog you cannot change it.

At the Chart tile group make the following selections:

Chart title	Bar Chart 2
Number of series (1-4)	2
Number of points (1-24)	12
Chart details	Configure chart

Note the number of series has been set to 2.

Click the Configure chart hyperlink:

Chart details	
OK	
Series information	
Series name 1	Actual
Series name 2	Budget
X-Axis labels	
Label 1	{Lab1}
Label 2	{Lab2}
Label 3	{Lab3}
Label 4	{Lab4}
Label 5	{Lab5}
Label 6	{Lab6}
Label 7	{Lab7}
Label 8	{Lab8}
Label 9	{Lab9}
Label 10	{Lab10}
Label 11	{Lab11}
Label 12	{Lab12}
X-Axis tooltips	
Tooltip 1	{Tip1}

Note that there are two series in this example – enter the series names as required.

Retain the X-Axis labels and X-Axis tooltip values.

Click on Edit Tile Chart.

We will be using the following custom SQL statement:

```
-- Actual
select
  1      as 'Series',
  10500 as 'P1',
  11000 as 'P2',
  12000 as 'P3',
  14000 as 'P4',
  13000 as 'P5',
  13000 as 'P6',
  16000 as 'P7',
  14500 as 'P8',
  15500 as 'P9',
  16500 as 'P10',
  16000 as 'P11',
  17000 as 'P12',
```

```

'Jan' as 'Lab1',
'Feb' as 'Lab2',
'Mar' as 'Lab3',
'Apr' as 'Lab4',
'May' as 'Lab5',
'Jun' as 'Lab6',
'Jul' as 'Lab7',
'Aug' as 'Lab8',
'Sep' as 'Lab9',
'Oct' as 'Lab10',
'Nov' as 'Lab11',
'Dec' as 'Lab12',
'January' as 'Tip1',
'February' as 'Tip2',
'March' as 'Tip3',
'April' as 'Tip4',
'May' as 'Tip5',
'June' as 'Tip6',
'July' as 'Tip7',
'August' as 'Tip8',
'September' as 'Tip9',
'October' as 'Tip10',
'November' as 'Tip11',
'December ' as 'Tip12'
--from
-- [Table] with (nolock)
--where
-- Column = @Customer

union

-- Budget
select
2 as 'Series',
10000 as 'P1',
10000 as 'P2',
10000 as 'P3',
10000 as 'P4',
12000 as 'P5',
12000 as 'P6',
12000 as 'P7',
12000 as 'P8',
14000 as 'P9',
14000 as 'P10',
14000 as 'P11',
14000 as 'P12',
'Jan' as 'Lab1',
'Feb' as 'Lab2',
'Mar' as 'Lab3',
'Apr' as 'Lab4',
'May' as 'Lab5',
'Jun' as 'Lab6',
'Jul' as 'Lab7',
'Aug' as 'Lab8',
'Sep' as 'Lab9',
'Oct' as 'Lab10',
'Nov' as 'Lab11',
'Dec' as 'Lab12',
'January' as 'Tip1',
'February' as 'Tip2',
'March' as 'Tip3',
'April' as 'Tip4',
'May' as 'Tip5',
'June' as 'Tip6',
'July' as 'Tip7',
'August' as 'Tip8',
'September' as 'Tip9',
'October' as 'Tip10',
'November' as 'Tip11',
'December ' as 'Tip12'
--from
-- [Table] with (nolock)
--where
-- Column = @Customer
order by
Series

```

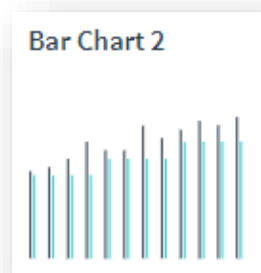
The first series is the same as the previous example.

Note however that there is a 'union' clause that joins two separate SQL select statements. The second statement defines series 2 and contains different data points.

Also, as we have two series, we must ensure that the first row represents series 1 and the second row represents series 2 – this is handled by the 'order by Series' clause at the end of the SQL statement.

Important: The X-Axis labels returned by columns 'Tip1' thru 'Tip12' will be the same for both series.

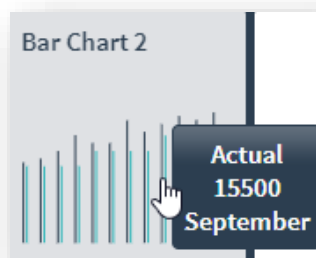
Once added to the user's workspace the tile will appear as follows:



Considerations:

- The tile title 'Bar Chart 2' is shown at the top of the tile
- The data points 'P1' thru 'P12' are shown as the height of a bar on the chart
- There are two series – one returned as a set of data points from each SQL row

You can mouse-over the data points to see more information:



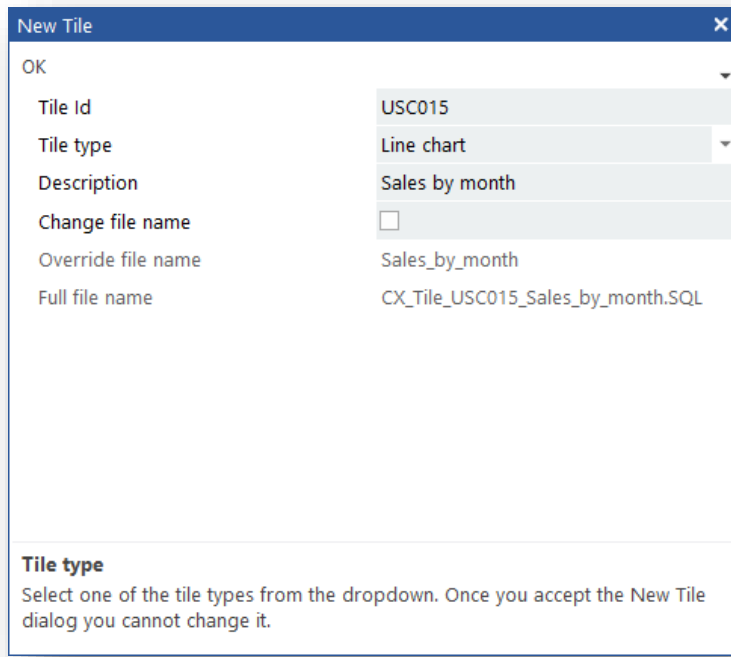
Considerations:

- The phrase 'Actual' is the series name from the 'configure chart' dialog
- The data value 15500 is the value returned from SQL column 'P9' row 1
- The phrase 'September' is the X-Axis tooltip return from SQL column 'Tip9'

CREATING A LINE CHART TO VIEW SALES

In this example we are going to create a line chart showing the last 24 months sales for a selected customer.

Start adding a new tile and select a Tile type: Line chart

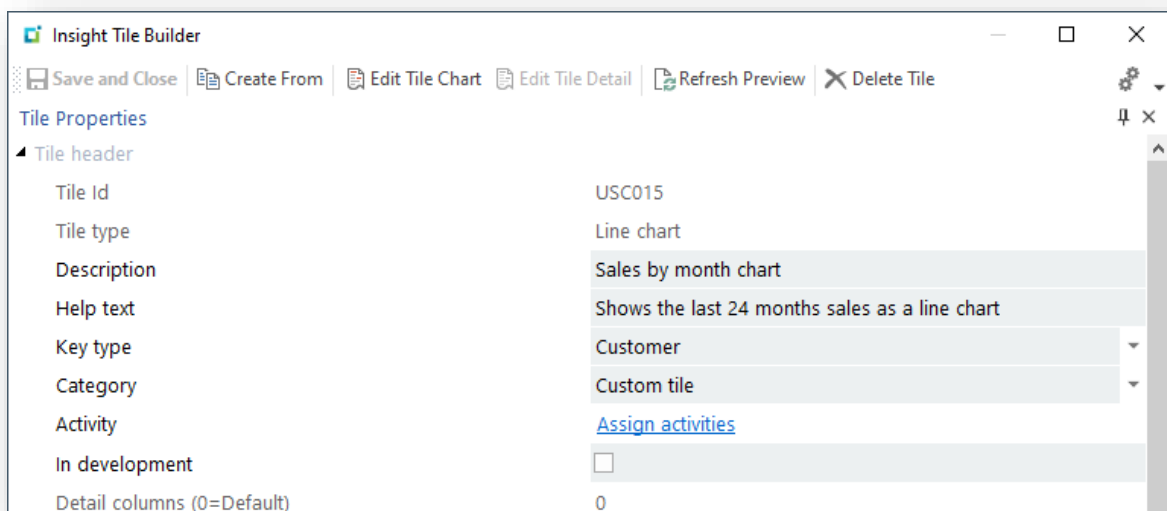


The 'New Tile' dialog box is shown with the following fields:

OK	
Tile Id	USC015
Tile type	Line chart
Description	Sales by month
Change file name	<input type="checkbox"/>
Override file name	Sales_by_month
Full file name	CX_Tile_USC015_Sales_by_month.SQL

Tile type
Select one of the tile types from the dropdown. Once you accept the New Tile dialog you cannot change it.

Click OK to start defining the tile properties:



The 'Insight Tile Builder' interface is shown with the following properties:

Tile Id	USC015
Tile type	Line chart
Description	Sales by month chart
Help text	Shows the last 24 months sales as a line chart
Key type	Customer
Category	Custom tile
Activity	Assign activities
In development	<input type="checkbox"/>
Detail columns (0=Default)	0

We have changed the description to 'Sales by month chart' to make it easier to find this tile later.

We have entered some help text to describe the purpose of the tile more fully. This becomes more useful as you start adding many custom tiles.

We have selected 'Customer' against the Key type dropdown. This will allow us to select data for a specific customer in context from the application.

Open the Chart tile group and make the following entries:

Chart tile	
Chart title	Sales by month
Number of series (1-4)	1
Number of points (1-24)	24
Chart details	Configure chart

Note that the number of series is 1 and number of data points is 24.

Click on the Configure chart hyperlink.

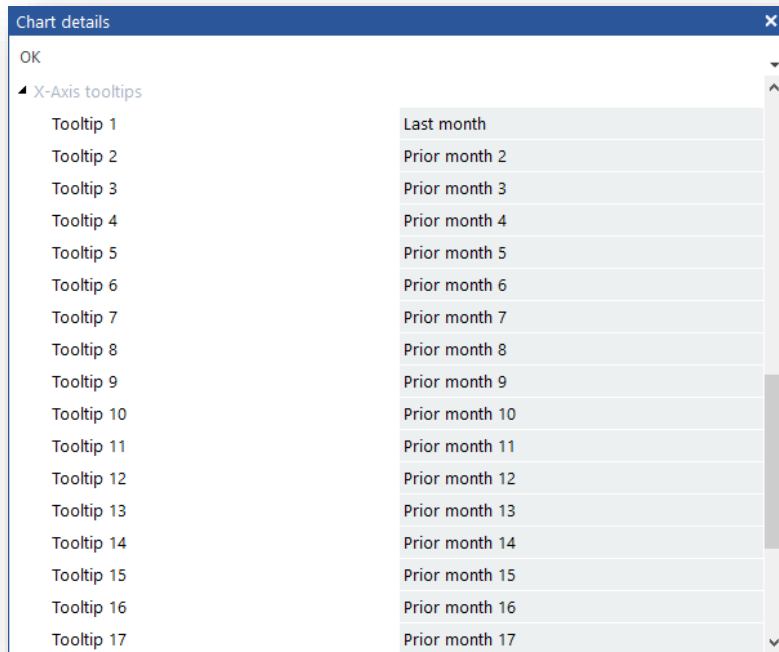
Chart details	
OK	
Series information	
Series name 1	Sales
X-Axis labels	
Label 1	1
Label 2	2
Label 3	3
Label 4	4
Label 5	5
Label 6	6
Label 7	7
Label 8	8
Label 9	9
Label 10	10
Label 11	11
Label 12	12
Label 13	13
Label 14	14
Label 15	15

We will enter the series name 'Sales'.

You must have a unique non-blank X-Axis Label for each point to be shown. In this case we have simply entered a number from 1-24 for each of these labels. They are not shown but must be defined and each must be unique.

Tab down the dialog and enter hard coded strings for each of the X-Axis tooltips.

See example:



Note that we could have also entered **receiving variables** named **{Tip1}** thru **{Tip24}** and returned the tooltip in the SQL select statement – this is just slightly easier in this example as the X-Axis tooltip values do not change based on the selected data.

Next, you should edit the 'Edit Tile Chart' function, ensure that the Custom SQL Query dialog is selected and capture the following SQL statement.

```

--SQL=Preamble
declare @Customer varchar (#{Customer.Length})
set @Customer = '#{Customer}'

--SQL=Select
select
  1 as 'Series'
  ,SalesValMth1 as 'P1'
  ,SalesValMth2 as 'P2'
  ,SalesValMth3 as 'P3'
  ,SalesValMth4 as 'P4'
  ,SalesValMth5 as 'P5'
  ,SalesValMth6 as 'P6'
  ,SalesValMth7 as 'P7'
  ,SalesValMth8 as 'P8'
  ,SalesValMth9 as 'P9'
  ,SalesValMth10 as 'P10'
  ,SalesValMth11 as 'P11'
  ,SalesValMth12 as 'P12'
  ,SalesValMth13 as 'P13'
  ,SalesValMth14 as 'P14'
  ,SalesValMth15 as 'P15'
  ,SalesValMth16 as 'P16'
  ,SalesValMth17 as 'P17'
  ,SalesValMth18 as 'P18'
  ,SalesValMth19 as 'P19'
  ,SalesValMth20 as 'P20'
  ,SalesValMth21 as 'P21'
  ,SalesValMth22 as 'P22'
  ,SalesValMth23 as 'P23'
  ,SalesValMth24 as 'P24'
from
  ArCustomerBal
where
  Customer = @Customer
  
```

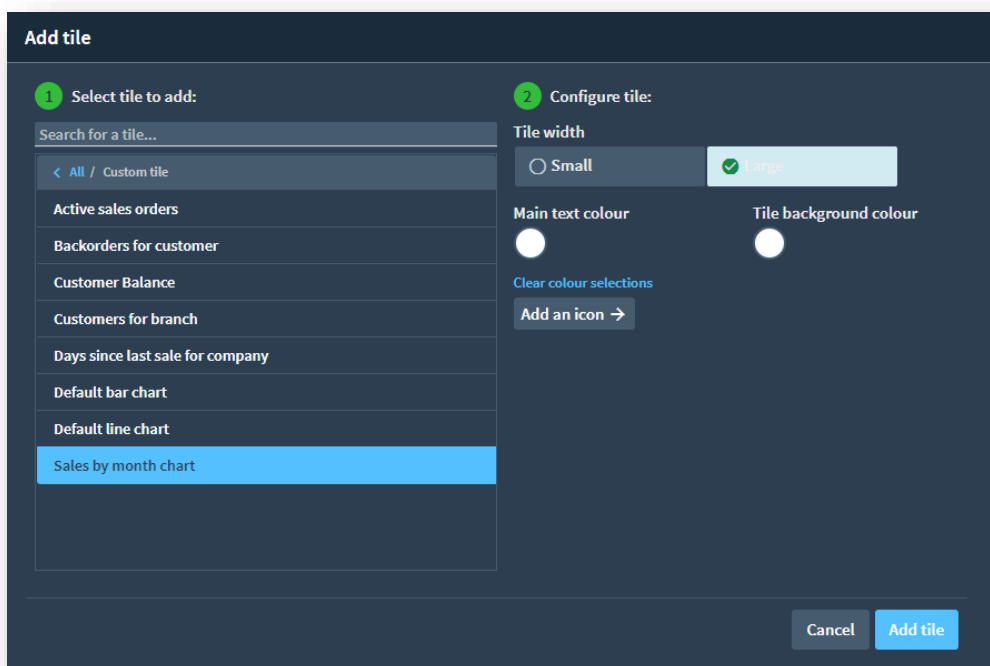
The SQL statement is:

```
--SQL=Preamble
declare @Customer varchar(${Customer.Length})
set @Customer = '$Customer'

--SQL=Select
select
  1 as 'Series'
  ,SalesValMth1 as 'P1'
  ,SalesValMth2 as 'P2'
  ,SalesValMth3 as 'P3'
  ,SalesValMth4 as 'P4'
  ,SalesValMth5 as 'P5'
  ,SalesValMth6 as 'P6'
  ,SalesValMth7 as 'P7'
  ,SalesValMth8 as 'P8'
  ,SalesValMth9 as 'P9'
  ,SalesValMth10 as 'P10'
  ,SalesValMth11 as 'P11'
  ,SalesValMth12 as 'P12'
  ,SalesValMth13 as 'P13'
  ,SalesValMth14 as 'P14'
  ,SalesValMth15 as 'P15'
  ,SalesValMth16 as 'P16'
  ,SalesValMth17 as 'P17'
  ,SalesValMth18 as 'P18'
  ,SalesValMth19 as 'P19'
  ,SalesValMth20 as 'P20'
  ,SalesValMth21 as 'P21'
  ,SalesValMth22 as 'P22'
  ,SalesValMth23 as 'P23'
  ,SalesValMth24 as 'P24'
from
  ArCustomerBal
where
  Customer = @Customer
```

Note that the SQL Preamble is used to retrieve the current customer in context – the same way as we have done when defining a text tile.

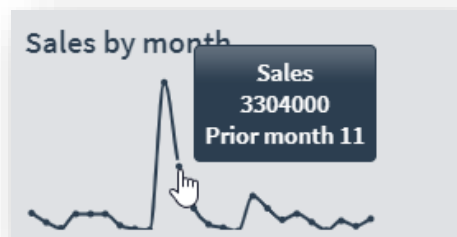
When adding the tile to the user's workspace we're going to select the 'Large' tile width.



Once added to the user's workspace, when selecting a customer, the past 24 months sales are shown as a line chart. An example is shown below:



Moving the mouse over an item shows the values as a tooltip:



ADDITIONAL NOTES ABOUT CHART TILES

SQL STATEMENTS AND MULTI-SERIES CHARTS

When you have a chart with multiple series, you must return multiple rows of data, one row for each series. There are several ways of doing this with a custom SQL statement.

Examples include:

- Using a 'union' to merge two independent SQL select statements
- Using a 'CTE' to create a data set from multiple SQL data sources
- Using a simple 'select' where the data is already separated into rows and can be ordered in the same sequence as the chart series definitions

In all cases you must remember to ensure that the rows are returned in the same sequence as your chart series definitions. Row 1 is series 1, row 2 is series 2 etc.

DIFFERENCES BETWEEN CHART AND TEXT TILES

The following table lists some of the differences in features between text and chart tiles.

Feature	Text tile	Chart tile
Detail drilldown	Supported	Not applicable
KPI definition	Supported	Not applicable
Tile preview (in Tile Builder)	Supported	Not applicable
Generated SQL (in Tile Builder)	Supported	Not applicable

Insight Tile Targets and KPIs

Insight Tiles are an effective way of presenting summary information that can be viewed at-a-glance. This together with the drilldown capability to see the detail can provide the user with information to help make effective business decisions.

It would be even more useful if you were able to provide thresholds that can be used to highlight where a tile has approached or exceed pre-determined values.

For example, showing the average age of invoices is useful. Being able to highlight when the average age is over (say) 90 days allows proactive credit management.

The target could be a system or company-wide goal or a specific agreed target for an individual user or all users who belong to a role within the organization.

The **Insight Tile** system provides the ability to set warning and/or critical threshold limits so that if a value exceeds these limits, the appearance of text tiles will be changed to highlight the situation.

Often businesses set **Key Performance Indicators** (KPIs) to help direct and manage personnel.

The ability to set thresholds against an **Insight Tile** uses this concept of KPIs. You will see the acronym **KPI** being used for this purpose.

KPI ATTRIBUTES

If you define an **Insight Tile** KPI, at run time the tile value is compared to pre-defined thresholds and if the value exceeds the thresholds, one or more of the following tile attributes will be changed and used instead of the original attributes:

- Main value text color
- Tile background color
- Optional icon and icon color

When used appropriately, changing the visual appearance of the tile, allows the viewer to clearly see that a tile value has exceeded a pre-determined threshold – allowing them to take appropriate action.

KPI GOAL TYPES

When defining an **Insight Tile** KPI there are three types of goal – they are:

- Minimizing
- Maximizing
- Target value

KPI GOAL: MINIMIZING

When a tile has a goal of **Minimizing** it means that the ideal tile value must be as small as possible.

This will often be to reduce the value to zero – but can be towards some arbitrary value (positive or negative).

For example, the following hypothetical tiles are aimed at minimizing their tile values:

- Receivables days outstanding
- Backorders for a company
- Number of customers on hold for a branch

When defining a KPI for a **Minimizing** tile you will be able to define a Warning threshold and a Critical threshold.

If the tile value is above the Warning threshold, but below the Critical threshold, then a warning set of attributes can be applied to the tile. Similarly, if the tile value is above the Critical threshold then a critical set of attributes can be applied.

In all cases if the threshold values are not reached or exceeded then the default set of attributes configured when initially defining the tile will be used.

KPI GOAL: MAXIMIZING

When a tile has a goal of **Maximizing** it means that the ideal tile value must be as large as possible.

For example, the following hypothetical tiles are aimed at maximizing their tile values:

- Company profit
- Sales vs. budget percentage

When defining a KPI for a **Maximizing** tile you will be able to define a Warning threshold and a Critical threshold.

If the tile value is below the Warning threshold, but above the Critical threshold, then a warning set of attributes can be applied to the tile. Similarly, if the tile value is below the Critical threshold then a critical set of attributes can be applied.

In all cases if the threshold values are not reached or exceeded then the default set of attributes configured when initially defining the tile will be used.

KPI GOAL: TARGET VALUE

When a tile has a goal of a **Target Value** it means that the ideal tile value must tend towards some fixed value.

Using high and low thresholds you can cause this type of KPI to aim towards a range of values rather than a single value.

For example, the following hypothetical tiles are aimed at having a **Target Value** as their tile values:

- Oven Temperature
- Profit margin for Product class
- Customer balance as percentage of all customers

When defining a KPI for a **Target Value** tile you will be able to define a Warning threshold and a Critical threshold for both high and low values.

For example, if an oven's temperature is being measured then if the temperature is over a 'Critical high' threshold or below a 'Critical low' threshold then the Critical attributes will apply. Similar logic applies for 'Warning high' and Warning low' threshold values.

In all cases if the threshold values are not reached or exceeded then the default set of attributes configured when initially defining the tile will be used.

KPI THRESHOLD VALIDITY

As mentioned previously, depending on the KPI goal of **Minimizing**, **Maximizing** or **Target Value**, you can define one or more thresholds. All the thresholds are optional, but you should set at least one for a KPI to have any meaning.

The following table summarizes which thresholds are valid for each goal type.

Threshold	Minimizing	Maximizing	Target Value
Critical high	Valid		Valid
Warning high	Valid		Valid
Warning low		Valid	Valid
Critical low		Valid	Valid

The Critical and Warning high thresholds can be defined as greater-than (>) or greater-than-or-equal-to (>=). This will allow you to select whether the value itself will cause the threshold to have been 'reached' or whether it must be 'exceeded'.

Similarly, the Critical and Warning low thresholds can be defined as less-than (<) or less-than-or-equal to (<=).

KPI THRESHOLD COMPARISON – VALUE AND KPIVALUE

At run time, the value returned by the SQL column named **Value** is compared to the KPI thresholds to determine if a warning or critical target has been exceeded or not.

This will typically be the same value as shown to the user on the text tile.

The only difference is that the tile definition may have an applied data type, including rounding, adjusting the value shown to the user.

For example, using the **AutoCurrency** data type with two decimals means that when an exact value of \$1,969,183.05 is returned, the user may be presented with: \$ 1.97M

The exact numeric value (\$1,969,183.05 in the above example) returned by the **Value** column is the numeric value used when comparing against one of the thresholds.

If required, the text tile summary SQL can return an alternate SQL column named **KpiValue**. In this case the exact numeric value returned by **KpiValue** will be used to compare against the warning and/or critical thresholds and not the column named **Value**.

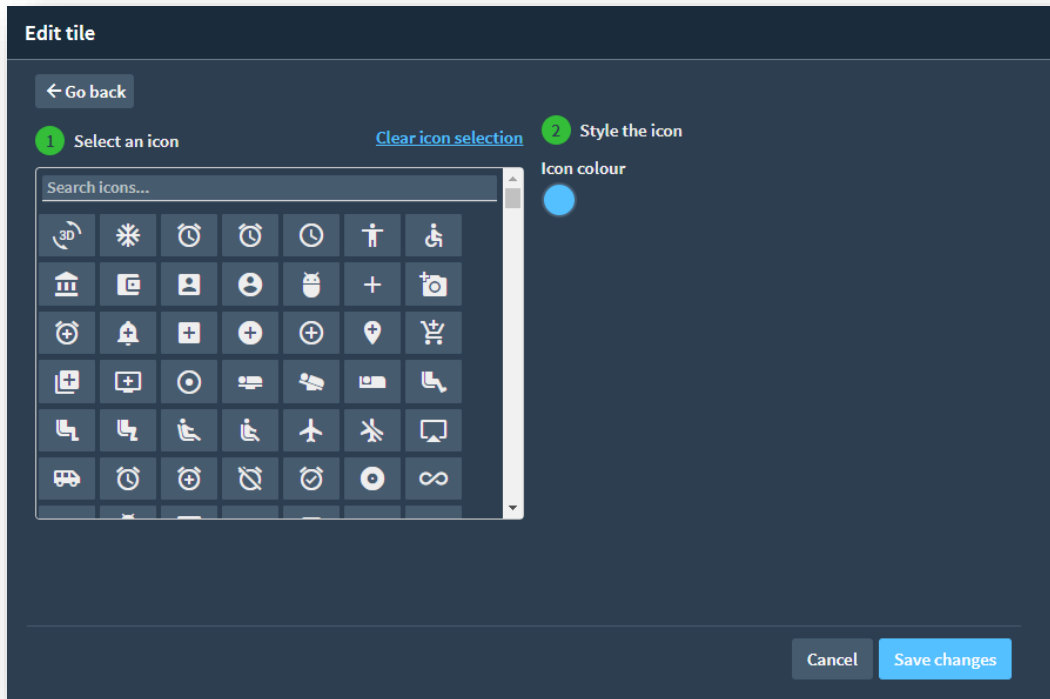
TILE ICONS AND COLORS

When you add a tile to a user's workspace, you can optionally specify an icon and the color for the icon.

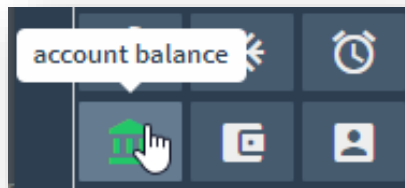
When defining a KPI for a tile and a threshold is exceeded you can override the icon and/or the icon color if required.

TILE ICON

When adding an icon, the following dialog is shown:

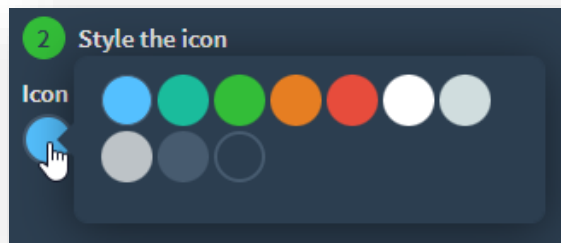


You can mouse-over to see the icon description. For example:













TILE ICON COLOR

You can then select from the standard color palette. The following is shown:



These standard palette colors each have a 'Color keyword' as shown in the table below:

Color keyword	RGB	Sample	Notes
Primary	#016699		Default icon color
Info	#33CCCC		
Success	#27AE60		
Warning	#F39C12		
Danger	#DD4B39		
Inverse	#222222		Default main text color
Gray	#999999		
LightGray	#DFDFDF		
VeryLightGray	#F6F6F6		
White	#FFFFFF		

You can assign an icon and color name to each of the applicable thresholds. You can elect not to change the icon and/or not to change the icon color, in which case the previous default (as configured when adding the tile to the user's workspace) will apply.

TILE COLORS

Like the icon color logic mentioned above, you can specify a color to be used to override each of the following when a threshold is reached.

- Tile background color
- Tile value text color

KPI LEVEL

When describing KPIs you describe the audience applicable for the specific KPI definition. This can be applied to one of the following:

1. System-wide
2. Selected Company
3. Selected Role
4. Selected Operator

The KPI Levels form a hierarchical definition and only one KPI definition (set of thresholds) will apply.

If there is an Operator KPI definition, then that set of thresholds will apply. However, if there is not an Operator KPI then a relevant Role KPI definition will apply. If there is neither an Operator nor Role KPI definition, then any company KPI definition will apply. Lastly, if none of the above KPI definitions exist then the system-wide KPI definition will apply.

KPI DEFINITION ACCESS CONTROL

A system administrator can add, change, and delete KPI definitions for all KPI levels. However, if required you can configure that specific operators can maintain various KPI levels.

Against each operator the following Tile KPI Access Control can be applied:

KPI Access Control	Description
KPI Administrator	Can configure any KPIs – administrators default to having this permission (when enabled none of the other KPI permissions are relevant)
Company supervisor	Can configure company KPIs – for Companies to which the operator has access
Role supervisor	Can configure Role KPIs - for Roles to which the operator has access
Operator self-service	Can configure Operator KPIs for themselves only

System administrators will default to 'KPI Administrator' Access Control enabled. Everyone else will default to all KPI Access Control disabled.


When adding an operator, you can limit which permissions are relevant to the user. See the topic: [KPI Permissions at the user level](#)

EXAMPLE: STANDARD TILE – AVERAGE DAYS TO PAY FOR BRANCH

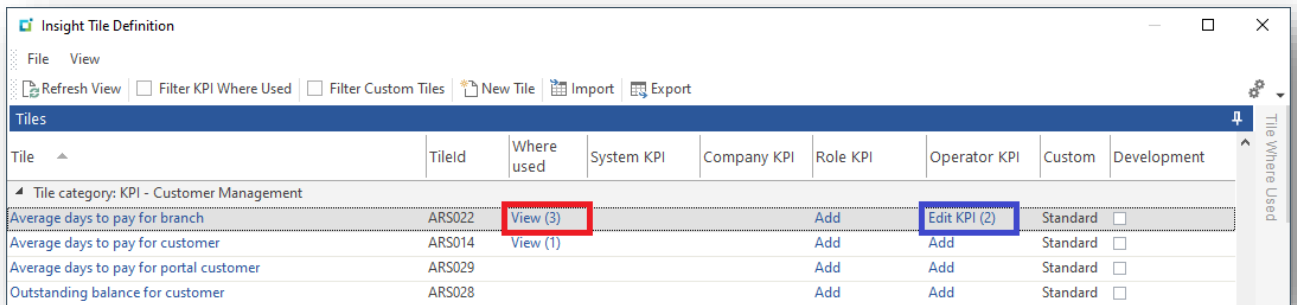
The standard text tile that shows the 'Average days to pay for a branch' (ARS022) will be used to demonstrate configuring a KPI.

In this example, the following will be configured:

Goal type: **Minimizing**

Severity	Threshold	Color	Icon	Image
Critical high	>=120	Danger	Error	
Warning high	>=60	Critical	Warning	

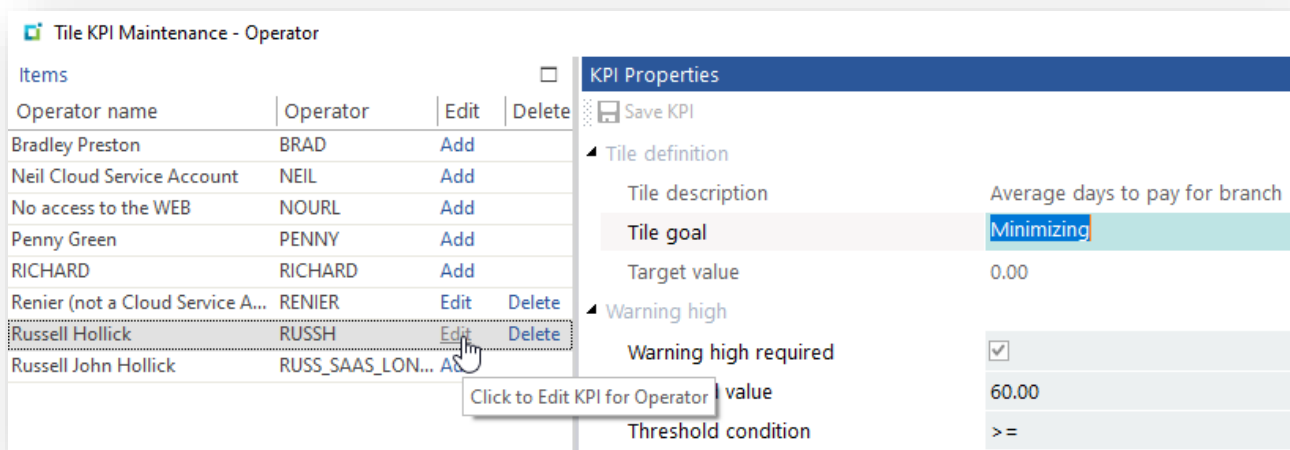
Load the Insight Tile Definition application (IMPKPI) and locate the tile to be maintained.



For your information the 'Where used' column allows you to see where the **Insight Tile** has been assigned to a user's workspace. This allows you to see how many places will be affected if you edit the tile or set KPIs. See the red highlight above.

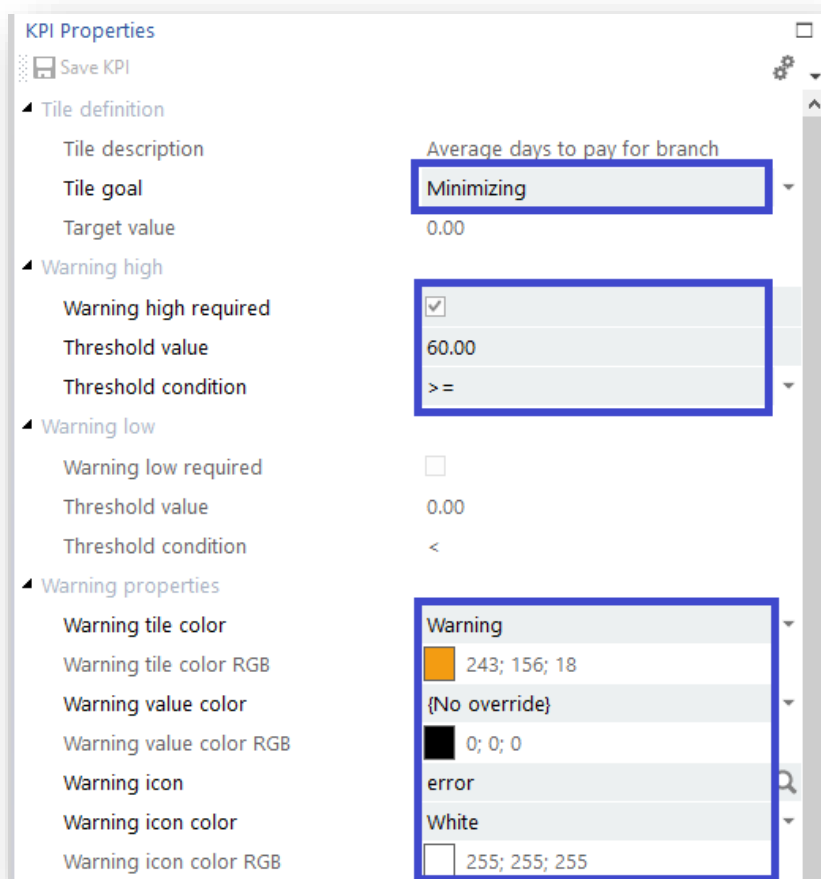
In this example we are going to set a KPI for a specific operator. Click on the hyperlink (highlighted in blue above). You will be taken to the Tile KPI Maintenance – Operator dialog.

See example below:



In this test system, the operators that the current user can edit are shown and we can Add, Edit or Delete a KPI as required.

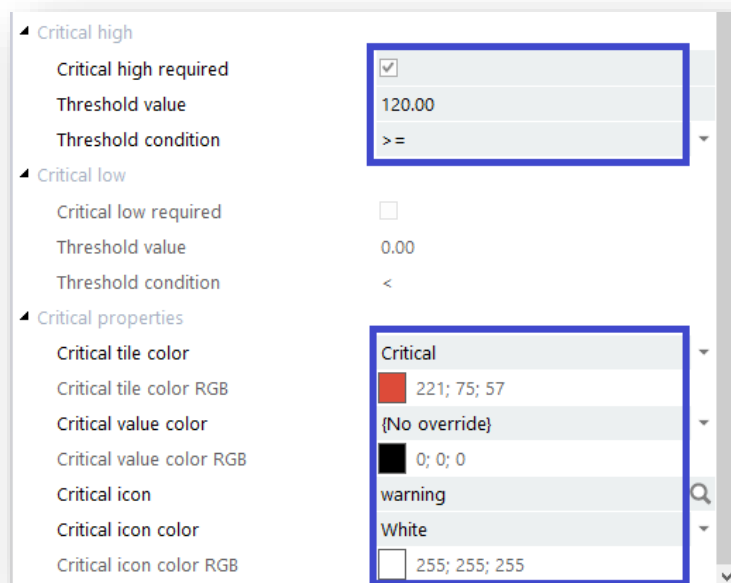
Select to Add/Edit to enable the KPI Properties for editing. Configure the KPI Properties as shown below:



Considerations:

- Tile goal is set to **Minimizing**
- Warning high threshold is checked
- Warning threshold is set to 60.00 and the condition '>='
- Warning properties
 - Warning tile color set to 'Warning'
 - Warning icon set to 'error'
 - Warning icon color set to 'White'

Scroll down to set the Critical properties:



Considerations:

- Critical high threshold is checked
- Critical threshold is set to 120.00 and the condition '>='
- Critical properties
 - Critical tile color set to 'Critical'
 - Critical icon set to 'warning'
 - Critical icon color set to 'White'

Save the KPI properties and return to the Insight Tile Definition application.

The KPI will now take effect the next time that a user logs into SYSPRO and runs an application where the tile has been added to the user's workspace.

In the table below we have provided some sample images of the various tile appearances at run time. The first row shows the tiles where we have not defined an override icon (this is quite clean), the second line shows where the icon override has been specified as per the examples above.

Normal	Warning	Critical

You will note that the 'Normal' tile (where no KPI threshold has been exceeded) was not originally defined with an Icon. However, you could have assigned a default icon and color when adding the tile to the user's workspace.

TILE AND ICON COLOR KEYWORDS

It is recommended that you select one of the colors from the SYSPRO color palette using one of the 'keywords' rather than specific a specific RGB color definition. The colors selected from the SYSPRO color palette will look more consistent with the rest of the SYSPRO user experience.

If you use the keywords, then when selecting an alternate theme, you can be sure that an appropriate color is selected. If you hard code the colors then if the user later changes their theme, the colors may not be appropriate.

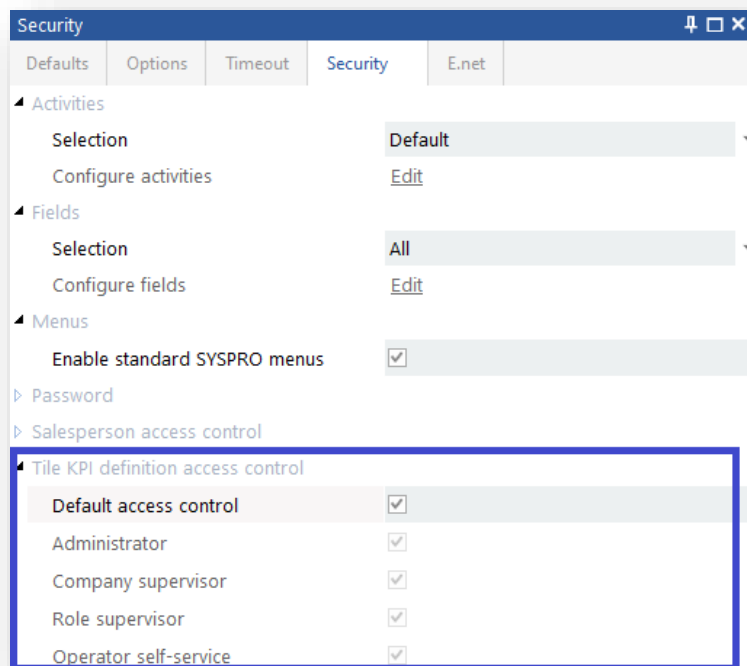
KPI PERMISSIONS AT THE USER LEVEL

As KPIs are used to highlight when warning and/or critical thresholds have been exceeded it's important to ensure that the correct thresholds are configured and are not changed inappropriately.

Important: Failure to restrict who can set KPIs can have an adverse effect on the business.

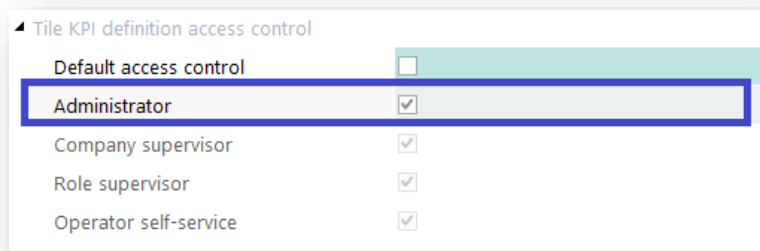
OPERATOR MAINTENANCE

The operator maintenance application provides Tile KPI definition access control options within the Security Tab.



The default is that anyone who has access to the Tile Definition program (IMPkpi) can add, change, or delete KPIs. This option is for backwards compatibility with earlier versions of SYSPRO 8 (prior to SYSPRO 8 2018 R2).

If you deselect the 'Default access control' checkbox, it selects the 'Administrator' option:



A Tile KPI Administrator has full control to add, change and delete Tile KPIs for the System and all Companies, Roles and Operators.

In effect, this is the same as the 'Default access control' that was pre-selected. However, the purpose of this option is to allow you to explicitly assign 'Administrative Tile KPI Definition' privileges to this operator – rather than having been 'inherited' the default access control before this assignment.

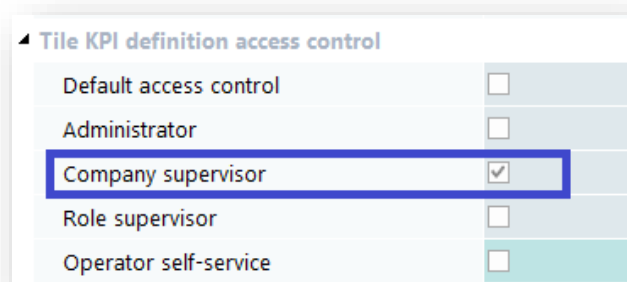
Deselecting the 'Administrator' access control option allows you to toggle each of the remaining three access control settings:

- Company supervisor
- Role supervisor
- Operator self-service

Each of these can be independently toggled on/off as required. These access control options are explained in detail below.

TILE KPI DEFINITION: COMPANY SUPERVISOR

The Company supervisor access control option allows the operator to maintain Tile KPI Definitions for companies to which they have access.

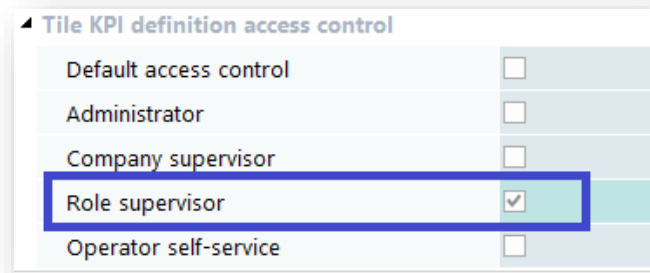


For example, if the operator has access to all companies then this operator can maintain Tile KPI definitions for all companies.

If the operator has an allowed or denied list of companies then the operator can maintain only those companies to which they have been granted access.

TILE KPI DEFINITION: ROLE SUPERVISOR

The Role supervisor access control option allows the operator to maintain Tile KPI Definitions for roles to which they belong and Tile KPI Definitions for all operators who belong to those roles.

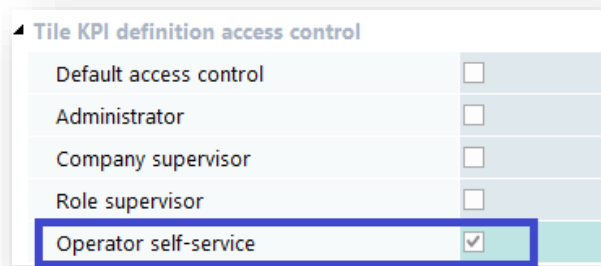


For example, if the operator belongs to three roles then this operator can maintain Tile KPI definitions for those three roles.

In addition, the operator can edit the Tile KPI definitions for all operators who belong to any of those three roles

TILE KPI DEFINITION: OPERATOR SELF-SERVICE

The Operator self-service access control option allows the operator to maintain their own 'personal' Tile KPI Definitions.



This allows the operator to edit tile definitions for their own operator only.

This can be useful when you wish the operator to be self-administering.

Insight Tile Import and Export

Creating and maintaining **Insight Tiles** so that users can see key values that allow them to make business decisions is extremely useful.

Whilst you can develop **Insight Tiles** on a live site, in some cases this is not practical.

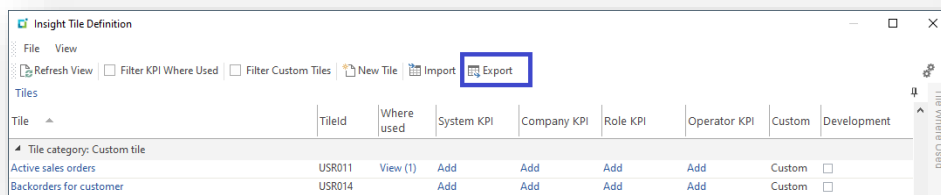
It can be more convenient to use a separate test system, or alternatively develop tiles on a separate development or support specific instance of SYSPRO.

In this case the ability to export tiles created on a separate SYSPRO instance and import them on the live target system is an important element of **Insight Tile** management.

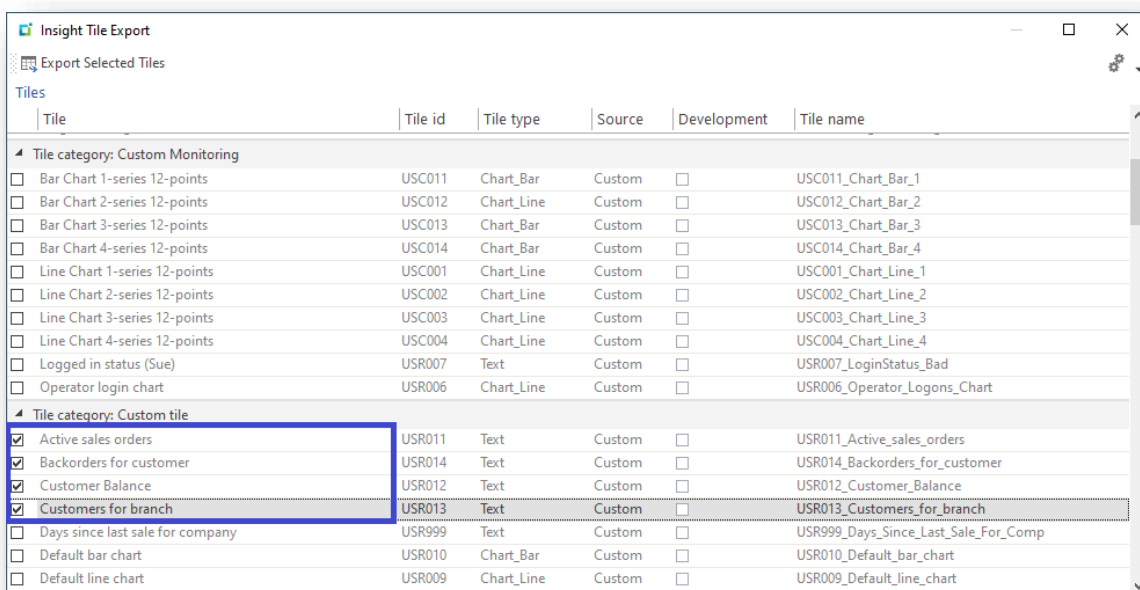
The Insight Tile Definition application has an export and import function – in both cases you can work with separate (individual) tile definition files (plain text files in a structured format) or a group of these collected into a single ZIP file.

TILE EXPORT

Load the Insight Tile Definition application (IMPKPI) and select the 'Export' toolbar button.



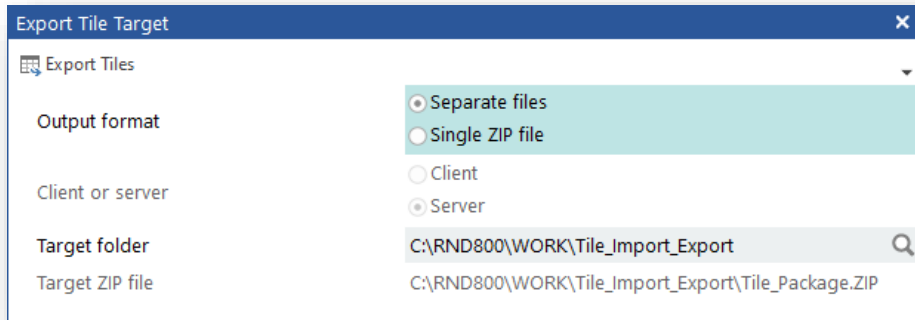
You will be presented with the Insight Tile Export dialog.



Select one or more tiles to be exported. You will typically be selecting custom tiles – i.e. tiles that you have created.

Note: You can elect to export standard SYSPRO tiles, however you cannot import these later.

Once you have selected one or more tiles, click the 'Export Selected Tiles' toolbar button. You will be presented with the Export Tile Target dialog:



If you wish to work with individual tile definition files (plain text files) then select 'Separate files', else select 'Single ZIP file'. It is often more convenient to work with a single ZIP file.

If you are running on a client server environment, you can select whether to export the file(s) to the client where you will typically have access or to the application server, where you may not have access.

Lastly, select the target folder when working with separate files or a ZIP file name.

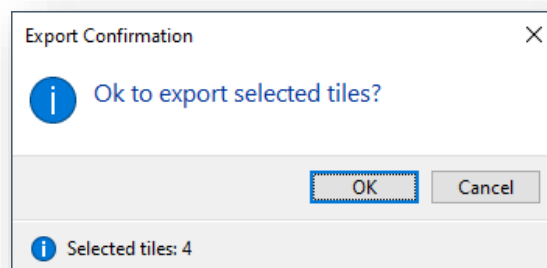
The default location will be a folder named **Tile_Import_Export** under your WORK folder. You can change this as required.

Click 'Export Tiles' to start the export function.

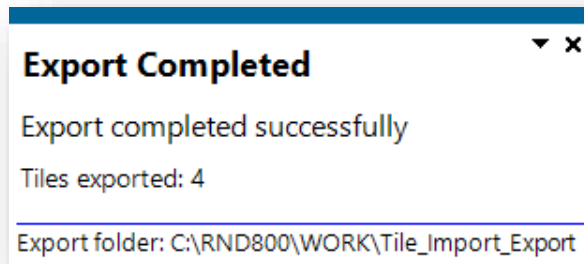
EXPORTING SEPARATE FILES

The export will not attempt to detect if you are about to overwrite a previous file or files in the selected folder.

You will be asked to start the export. It shows the number of selected tiles for confirmation.



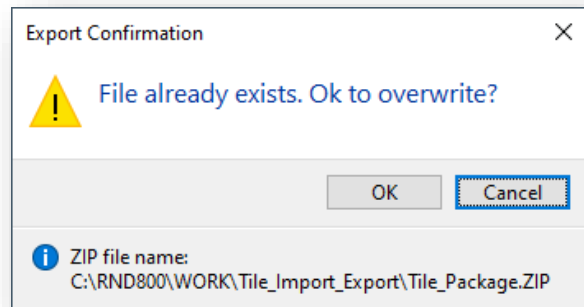
Upon completion you will receive a confirmation message showing how many individual tile definition files were exported and the folder location.



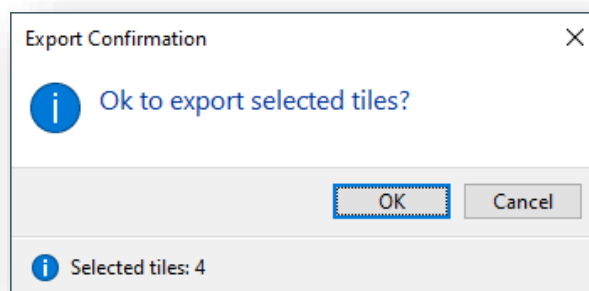
You will be returned to the Insight Tile Definition application.

EXPORTING ZIP FILES

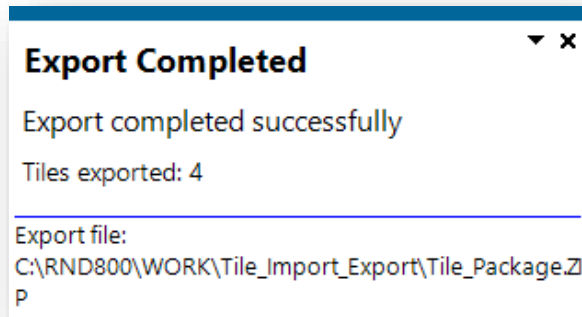
If the ZIP file already exists, you will be prompted to overwrite the file.



If you continue, you will be asked to start the export. It shows the number of selected tiles for confirmation.



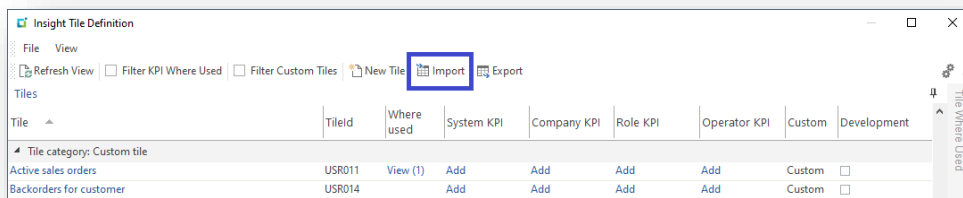
Upon completion you will receive a confirmation message showing how many tiles were exported and location of the ZIP file.



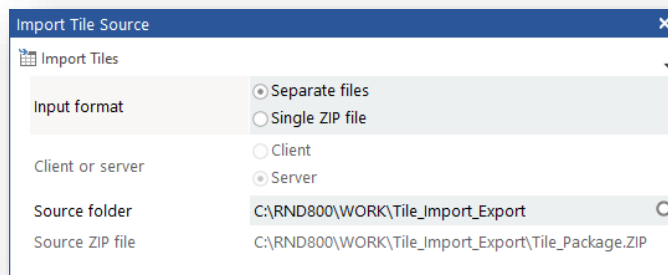
You will be returned to the Insight Tile Definition application.

TILE IMPORT

Load the Insight Tile Definition application (IMPKPI) and select the 'Import' toolbar button.



You will be presented with the Import Tile Source dialog.



If you chose 'Separate files', then you must define the 'Source folder'. When you click 'Import Tiles' the list of individual text files will be presented for further selection. Individual text files must match the pattern

- CX_Tile_??????.*.TXT

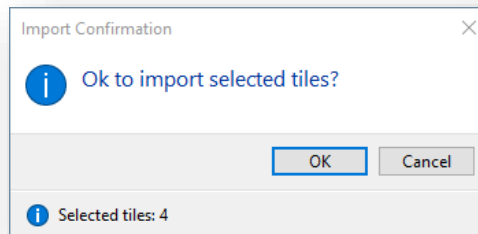
Alternatively chose 'Single ZIP file' and provide the name of the ZIP file. When you click 'Import Tiles' the file will be unzipped, and the list of tiles will be presented for further selection.

Tile	Tile id	Tile type	Source	Development	Tile file name
Tile category: Custom tile					
<input checked="" type="checkbox"/> Active sales orders	USR011	Text	Custom	<input type="checkbox"/>	CX_Tile_USR011_Active_sales_orders.SQL
<input checked="" type="checkbox"/> Customer Balance	USR012	Text	Custom	<input type="checkbox"/>	CX_Tile_USR012_Customer_Balance.SQL
<input checked="" type="checkbox"/> Customers for branch	USR013	Text	Custom	<input type="checkbox"/>	CX_Tile_USR013_Customers_for_branch.SQL
<input checked="" type="checkbox"/> Backorders for customer	USR014	Text	Custom	<input type="checkbox"/>	CX_Tile_USR014_Backorders_for_customer.SQL

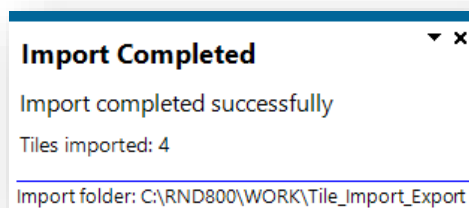
Once you have been presented with a list of tiles to be imported you can make your selection.

Note: If a tile with the same name already exists in the SYSPRO database, a warning will be shown. Continuing to the import will overwrite any previous tile with this name.

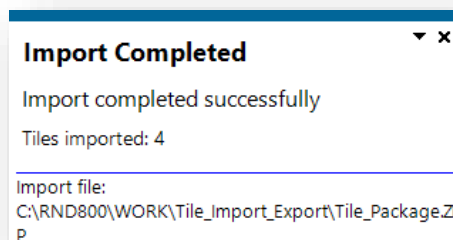
Click 'Import Selected Tiles' to start the import process. You will be asked to start the export. It shows the number of selected tiles for confirmation.



Upon completion you will receive a confirmation message showing how many tiles were exported. If you were importing separate files from a folder then the following will be shown:



Alternatively, when importing from a ZIP file you will see the following:



Upon completion you will be returned to the Insight Tile Definition application.

TILE IMPORT AND EXPORT NOTES

STANDARD INSIGHT TILES

You must not change standard **Insight Tiles**.

You can export standard **Insight Tile** definitions but cannot import them.

The Insight Tile Definition and Insight Tile Import applications do not allow you to save or import standard tiles.

UNIQUE TILE IDS

Important: All Tile Ids must be unique at the site.

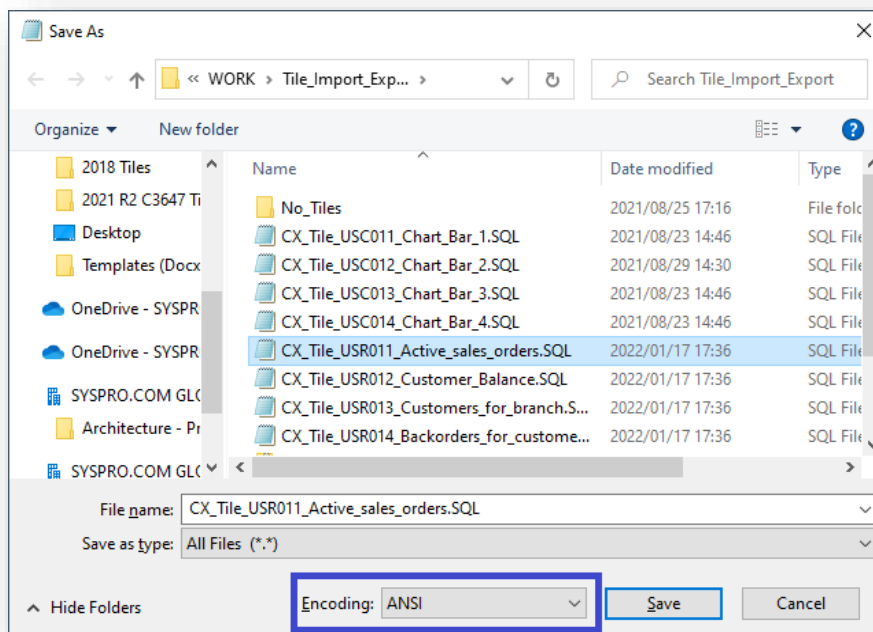
The Tile id is used to save internal preferences. For example, when using the detail drilldown, the column widths, sequence, and other preferences use the Tile id to uniquely identify each tile.

Due to this it is recommended that you use a 3-character prefix that uniquely defines your organization and does not conflict with one of the standard SYSPRO tiles.

SEPARATE TILE DEFINITION FILE FORMAT – ENCODING: ANSI

If you use the Insight Tile Builder to create and maintain tiles and subsequently export them as separate plain text definition files you could edit these using any text editor.

It's important to ensure that if using a tool such as Notepad that you select the 'Encoding: ANSI' when using the Save As dialog.



Appendix 1 – List of Key Types

The following key type values can be used wherever a 'Key Type' is required in a tile definition.

Key type	Key description	Length	Format
Financial keys			
Customer	Customer	15	Yes
CustomerBranch	Customer branch	10	
CustomerClass	Customer class	10	
Salesperson	Salesperson	20	
Area	Customer area	10	
CommissionCode	Commission code	2	
InvoiceTerms	AR invoice terms	2	
BuyingGroup	Buying group	10	Yes
Contact	Contact name	50	
Account	Contact management account code	15	
CustomerInvoice	Customer invoice number	20	Yes
Supplier	Supplier	15	Yes
SupplierBranch	Supplier branch	10	
SupplierClass	Supplier class	10	
PaymentRun	AP Payment run cycle	15	Yes
Bank	Bank	15	
LedgerCode	GL code	35	
StatisticalGLCode	Statistical GL code	35	
Asset	Asset code	30	Yes
AssetLocation	Asset location	10	
AssetType	Asset type	4	
AssetBranch	Asset branch	10	
AssetCostCenter	Asset cost center	20	
AssetGroup	Asset group	10	
AssetOwner	Asset owner	10	
AssetStatus	Asset status	10	
Mechanic	Mechanic	20	
CapexItem	Capex item	15	Yes
CapexType	Capex type	6	
CapexClassification	Capex classification	6	
CapexJob	Capex job	20	
TariffCode	Tariff code	15	
Distribution keys			
StockCode	Stock code	30	Yes

Key type	Key description	Length	Format
ProductClass	Product class	20	
Buyer	Buyer	20	
Planner	Planner	20	
DrawingNumber	Drawing number	32	
Warehouse	Warehouse	10	
SalesOrder	Sales order number	20	Yes
OrderType	Sales order type	2	
DispatchNote	Dispatch note	20	Yes
Rma	RMA	20	Yes
GitReference	Goods in transit reference	20	Yes
Requisition	Requisition number	10	Yes
PurchaseOrder	Purchase order number	20	Yes
Grn	Good received notes	20	Yes
Manufacturing keys			
Job	Job number	20	Yes
JobClassification	Job classification	10	
Manufacturer	Manufacturer	30	
Lot	Lot number	50	Yes
Quotation	Quotation	20	
CostCenter	Cost center	20	
WorkCenter	Work center	20	
Machine	Machine	30	
Employee	Employee	20	
Resource	Resource	30	
System-wide keys			
Company	Company id	4	
Nationality	Nationality	3	
Currency	Currency	3	
TaxCode	Tax code	3	
GstCode	Canadian GST code	3	
Operator	Operator	20	
Role	Operator role	3	



The **Length** field indicates the maximum length of variable required to store the key on a table.

- For example: Customer = 15, StockCode = 30
- This allows you to include a key length variable in your SQL statements such as:
 - `declare @Customer varchar(${Customer.Length})`
- This will be replaced with
 - `declare @Customer varchar(15)`

Keys with the **Format** column set to 'Yes' will be formatted according to SYSPRO key definitions. Other values will be returned 'as is'.

Appendix 2 – Data types

The following data types can be used when defining **receiving variables** use for summary text tiles.

Datatype	Notes	Decimals allowed	Example	Max size (int.dec)
String	Arbitrary alphanumeric string	N/A	Bayside Bikes	200
AutoCurrency	Currency value abbreviated	Yes	\$ 38M	20.6
AutoNumber	Number abbreviated	Yes	23K	20.6
Integer	Whole number field	No	1234	20.0
Percentage	Edit using SYSPRO numeric editing with '%' appended.	Yes	12%	12.6
Number	Edit using SYSPRO numeric editing	Yes	100 000,00	12.6
Value	Edit using SYSPRO value editing	No	38 ,012 891,00	12.2
Cost	Edit using SYSPRO cost editing	No	123 456,12345	12.5
Price	Edit using SYSPRO price editing	No	123 456,12345	12.5
Quantity	Edit using SYSPRO quantity editing	Yes	10 456,123456	12.6
Years	Number of years	Yes	1.7	20.6
Months	Number of months	Yes	18	20.6
Weeks	Number of weeks	Yes	26	20.6
Days	Number of days	Yes	16	20.6
Hours	Number of hours	Yes	6	20.6
Minutes	Number of minutes	Yes	59	20.6
Seconds	Number of seconds	Yes	30	20.6
DateLong	SYSPRO Long date format	N/A	September 30, 2022	50
DateShort	SYSPRO Short date format	N/A	2022/09/30	50
DateCYMD	Date in CCYY-MM-DD format	N/A	2022-09-30	10

DEFAULT DATA TYPES: ALPHANUMERIC – 'STRING'

When an alphanumeric SQL datatype is returned, the default formatting will be **String** – the value is returned 'as is'. Alphanumeric SQL datatypes supported are:

- Char
- Varchar
- Nchar
- Nvarchar

If you wish to return other alphanumeric SQL datatypes then use CONVERT or CAST to return one of the supported datatypes shown above.

DEFAULT DATA TYPES: NUMERIC – ‘AUTO NUMBER’

When a numeric SQL data type is returned, the default formatting applied will be **AutoNumber** with zero decimal places. This means that the number will use language code abbreviation logic to show the number. For example, 27,123 could be returned as ‘27K’.

Numeric SQL datatypes supported are:

- Decimal – Up to Decimal(26,6)
- Bigint
- Int
- Smallint
- Tinyint
- Real
- Float

If you wish to return other numeric SQL datatypes then use CONVERT or CAST to return one of the supported datatypes shown above.

For more information relating to the AutoNumber and AutoCurrency data types see [Appendix 4: AutoNumber and AutoCurrency](#).

DEFAULT DATA TYPES: DATE – ‘DATE LONG’

When a date SQL data type is returned, the default formatting applied will be **DateLong**. This means that the SYSPRO long date formatting will be used to present the date. The Long date format is defined against the System Setup which can be overridden using the Company setup.

Numeric SQL datatypes supported are:

- Datetime
- Smalldatetime

If you wish to return other date SQL datatypes then use CONVERT or CAST to return one of the supported datatypes shown above.

Appendix 3 – System variables - `#{Variable}`

All tile definition SQL statements can include system variables that provide access to many values relating to the current SYSPRO environment.

All system variables are named `#{variable}` where 'variable' is the variable name. You must use the exact case as shown here.

System Variable	Description	Max length	Example value
<code>#{Operator}</code>	Current operator code	20	ADMIN
<code>#{OperatorName}</code>	Current operator name	50	System Administrator
<code>#{OperatorEmail}</code>	Current operator email address	255	admin@bayside.com
<code>#{OperatorLanguage}</code>	Current operator language	2	EN
<code>#{OperatorLocation}</code>	Current operator location	50	Head Office
<code>#{OperatorSalesperson}</code>	Current operator salesperson	20	AG
<code>#{OperatorEccUser}</code>	Current operator ECC user	20	AG
<code>#{RequisitionUser}</code>	Current operator requisition user	20	AG
<code>#{DefaultApBranch}</code>	Current operator default AP branch	10	NY
<code>#{DefaultArBranch}</code>	Current operator default AR branch	10	NY
<code>#{DefaultBank}</code>	Current operator default bank	15	FB
<code>#{DefaultBuyer}</code>	Current operator default buyer	20	AG
<code>#{DefaultJobClass}</code>	Current operator default job classification	10	SPEC
<code>#{DefaultWarehouse}</code>	Current operator default wh	10	FG
<code>#{DefaultSupplier}</code>	Current operator default supplier	15	ACME01
<code>#{PortalSupplier}</code>	Operator defined Portal supplier	15	ACME01
<code>#{PortalCustomer}</code>	Operator defined portal customer	15	ACME01
<code>#{Company}</code>	Current company id	4	EDU1
<code>#{CompanyName}</code>	Current company name	50	The Outdoors Company
<code>#{CompanyReference}</code>	Current company reference	50	Outdoors Test Company
<code>#{CompanyDb}</code>	Current company database name	20	SysproEdu1
<code>#{SharedInventoryDb}</code>	Database for Inventory tables	20	SysproEdu1
<code>#{SharedGLDb}</code>	Database for GL tables	20	SysproEdu1
<code>#{SystemDb}</code>	System-wide database name	20	Sysprodb
<code>#{Currency}</code>	Local currency code	3	USD
<code>#{CurrencyDescription}</code>	Local currency description	50	US Dollar
<code>#{FinancialPeriods}</code>	No. of financial periods per year	2	12
<code>#{TopX}</code>	Top X items for detail	12	1000 (default=5000)

Appendix 4 – AutoNumber and AutoCurrency

The **AutoCurrency** data type uses the current language to look-up and abbreviate currency values using the lookup file IMPLNA.IMP. It uses the section between **[LangCurStart]** and **[LangCurEnd]**.

Similarly, the **AutoNumber** data type also uses a similar look-up in IMPLNA.IMP. It uses the section between **[LangNumStart]** and **[LangNumEnd]**.

The **AutoNumber** and **AutoCurrency** types work similarly using the IMPLNA.IMP definition. The following worked example helps to demonstrate how this logic is applied.

A fragment from IMPLNA.IMP is shown below:

```
; IMPLNA.IMP - List of SYSPRO Language Value Abbreviations
;
; Copyright (c) 1994-2017 SYSPRO Ltd. All rights reserved.
;
; @(#) Version : 8.0.000      Last updated : 2017/02/12 22:40
;
; Currency value abbreviation logic per language [LangCurStart] -> [LangCurEnd]
; Format tag (Frmt:) defines how to edit currency value:
;   Valid place holders are:
;     C - Currency code eg 'USD', 'EUR', 'R'
;     V - Currency value eg '7', '23', '150'
;     A - Abbreviation  eg 'K', 'M', 'B'
;     S - Single space
;
; Numeric value abbreviation logic per language [LangCurStart] -> [LangCurEnd]
; Format tag (Frmt:) defines how to edit any numeric value:
;   Valid place holders are:
;     V - Currency value eg '7', '23', '150'
;     A - Abbreviation  eg 'K', 'M', 'B'
;     S - Single space
;
; First two entries must be 'EN' English [LangCurStart] and then [LangNumStart]
;-----
[LangCurStart]
Lang:EN           ; English
Frmt:CSVA        ; eg 'USD 750K'
Fact:T:1000000000000 ; Trillion
Fact:B:1000000000  ; Billion
Fact:M:1000000     ; Million
Fact:K:1000       ; Thousand
[LangCurEnd]
;
[LangNumStart]
Lang:EN           ; English
Frmt:VA          ; eg '750K'
Fact:T:1000000000000 ; Trillion
Fact:B:1000000000  ; Billion
Fact:M:1000000     ; Million
Fact:K:1000       ; Thousand
[LangNumEnd]
```

Assume the currency value to be formatted is 27123 (twenty-seven thousand, one hundred and twenty three) and the current currency code is 'USD'.



The [\[LangCurStart\]](#) section will be parsed to determine the logic to be applied.

The value 27123 is divided by each 'Fact' entry until a value 1 or higher results. Thus 27123 divided by **1000000000000** is less than 1, as is each value until we get to **Fact:K:1000** in which case the integer value after dividing by **1000** is '27' and the string 'K' will be returned.

Then the formatting logic defined by **Fmt:CSVA** is inspected and the formatting applied as follows:

- C - First the currency code is used ('USD')
- S - Then a space
- V - Then the value calculated ('27')
- A - Then the abbreviated string ('K')

This will result in the string: **USD 27K**

With appropriate definition of dividing factors and abbreviation strings many different styles of currency value and numeric value formatting can be applied.

If required, you can supply a custom 'CUSLNA.IMP' file in the custom store folder:

- Plugin\CustomStore

The first entry in IMPLNA.IMP must be the English definition as shown in the example above.

You can override the Currency code and Language code in your tile definition file by defining column values with captions **{Override.Currency}** and/or **{Override.Language}** (exact case).

See the topic [Smart editing - AutoCurrency](#) for more details.

Appendix 5 – System Limits

This section describes some limits imposed on the tile architecture.

Maximum	Description
	Tile properties
60 chars	Max length of tile description
500 chars	Max length of tile help text (long description)
50 chars	Max length of tile category
200 chars	Max length of summary tile title string
200 chars	Max length of summary tile subtitle string
200 chars	Max length of summary tile value string
200 chars	Max length of summary tile footer string
	Parameter Information
10 params	Max number of parameters
30 chars	Max length of parameter name
100 chars	Max length of parameter description
	Chart information
4 series	Max number of chart series
24 points	Max number of chart points per series
100 chars	Max series name length for chart
	SQL limits
30 chars	Longest column name returned in 'as' clause in SQL SELECT
400 chars	Max length of where condition (generated SQL). No limit when custom SQL.
65,000 chars	Max length of single SQL batch command (after parameters expanded)
	Other limits
50 chars	Max icon name length (KPIs)
1,000 chars	Max length of single line in tile definition file
60 chars	Max length of tile definition name (excluding 'UX_Tile_' prefix and '.SQL' suffix) - Spaces not allowed



www.syspro.com

Copyright © SYSPRO. All rights reserved.
All brand and product names are trademarks or
registered trademarks of their respective holders.

