



Data Aggregation

Planon Software Suite

Version: L105

© 1997 - 2024 Planon. All rights reserved.

Planon and the Planon logo are registered trademarks of Planon Software Development B.V. or its affiliates. All other product and company names mentioned herein are trademarks or registered trademarks of their respective companies. Planon Software Development B.V., its affiliates and/or licensors own the copyright to all Planon software and its associated data files and user manuals.

Although every effort has been made to ensure this document and the Planon software are accurate, complete and up to date at the time of writing, Planon Software Development B.V. does not accept liability for the consequences of any misinterpretations, errors or omissions.

A customer is authorized to use the Planon software and its associated data files and user manuals within the terms and conditions of the license agreement between customer and the respective legal Planon entity as soon as the respective Planon entity has received due payment for the software license.

Planon Software Development B.V. strictly prohibits the copying of its software, data files, user manuals and training material. However, customers are authorized to make a back-up copy of the original CD-ROMs supplied, which can then be used in the event of data loss or corruption.

No part of this document may be reproduced in any form for any purpose (including photocopying, copying onto microfilm, or storing in any medium by electronic means) without the prior written permission of Planon Software Development B.V. No copies of this document may be published, distributed, or made available to third parties, whether by paper, electronic or other means without Planon Software Development B.V.'s prior written permission.

About this Document

Intended Audience

This document is intended for *Planon Software Suite* users.

Contacting us

If you have any comments or questions regarding this document, please send them to: support@planonsoftware.com.

Document Conventions

Bold

Names of menus, options, tabs, fields and buttons are displayed in bold type.

Italic text

Application names are displayed in italics.

CAPITALS

Names of keys are displayed in upper case.

Special symbols



	Text preceded by this symbol references additional information or a tip.
	Text preceded by this symbol is intended to alert users about consequences if they carry out a particular action in Planon.

Table of Contents

Data Aggregation – An Introduction.....	6
Configuration of Data Aggregation.....	8
Registering extensions.....	8
Configuring ContainerSX in Field Definer.....	8
Configuring ReportSX in Field Definer.....	12
Configuring CalculateSX in Field Definer.....	13
Data aggregation properties business object.....	15
Configuring Data aggregation properties.....	16
Aggregation properties BO for ReportSX and CalculateSX.....	16
General configuration.....	16
Variables to use in SQL.....	22
SQL for Data level.....	26
Data level.....	26
Details level.....	28
Aggregation of fields.....	32
Scheduling of ReportSX or CalculateSX.....	35
Scheduling.....	35
Scheduling process.....	35
Working with Data Aggregation Manager.....	36
ContainerSX.....	37
ReportSX.....	37
CalculateSX.....	38
Authorization in DAM.....	38
DAM implementation steps.....	38
Field descriptions.....	41
Aggregation data - fields.....	41

Aggregation details - fields.....42

Index.....44

Data Aggregation – An Introduction

This document describes the **Data Aggregation** solution extension.

Solution extensions (SX) are developed to extend the Planon ProCenter functionality. Planon ProCenter calls these extensions at defined moments during the execution of the program. Once installed and activated, a solution extension is an integral part of the Planon ProCenter software and works in a similar way as business rules or other functionality as provided by Planon.

Extensions are always triggered directly or indirectly by user actions (such as selecting, adding, updating, changing a status etc.). In Planon ProCenter, business rules often (though not always) manifest themselves in the form of dialog boxes that contain an error message, warning or confirmation. User authorization is not taken into account when business rules are executed.

The primary purpose of **Data aggregation** is to aggregate data from Planon, link it as a benchmark of a property. But it can also be used for all other kinds of data aggregations. The aggregated data can be used as management information.

Data aggregation contains three components.

ContainerSX

This SX creates a benchmark structure ('container') in which the aggregated data can be placed. This benchmark structure can contain the following levels:

- Definition level
- Data level

ReportSX

This SX calculates and places the aggregated data in the benchmark structure that was generated by ContainerSX. In this task, a third level can be added:

- Details level

CalculateSX

This SX calculates and place aggregated data in the same way as the ReportSX does, but not in the benchmark structure. Consequently, you can use it to fill in aggregated data in each Planon business object.

When the status of aggregation definition is changed to the configured status, the following steps are conducted:

- Data is filled in configured fields at the **Aggregation data** level. Fields are filled by the SQL queries defined in the **Data aggregation properties** BO.
- Sum fields are processed. If certain fields are configured for aggregation, the values of the children fields are added to the parent's fields. If a certain field is not configured to be updated, it will not be changed.

- Status of the aggregation definition is updated.
After a successful calculation of data, the status of the aggregation definition is also updated. If ReportSX processing is not successful, an error message is displayed. Also, logging of the run is performed in the **Comment** field configured for the aggregation definition.
- Cleanup of existing Aggregation details level records.
Before the records and fields of the Aggregation details level are filled, the existing records are deleted. The deletion is only performed on the **Aggregation details** level records that are not linked to an **Aggregation data** level record that is flagged for not updating.
- Aggregation details level records are added and fields are filled with data.
The Aggregation details level records are added and the fields are filled as configured in the **Data aggregation properties** BO.



For information on installing and registering an extension, see *Supporting data > Registering extensions*.

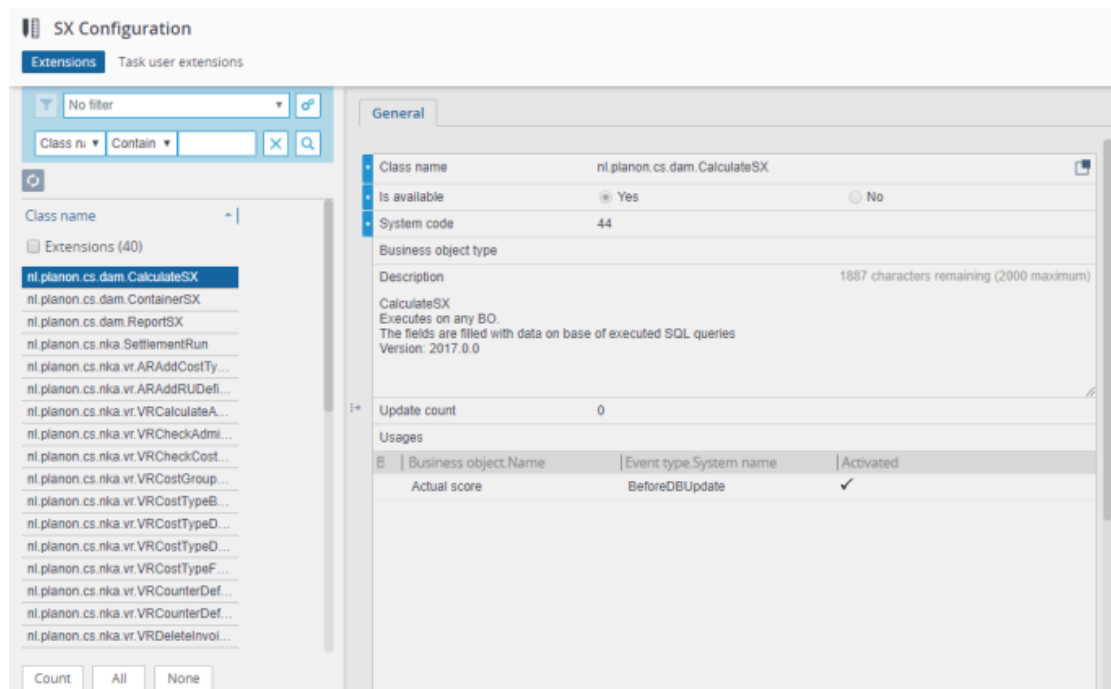
Configuration of Data Aggregation

This section describes how to configure the various Data Aggregation components

Registering extensions

Data Aggregation contains three extensions that together make the solution. An explanation of the usage of each extension can be found in the next chapter. To make the extensions available for use, they first have to be registered in Business processes > SX Configuration. The extensions can be added with the following class names:

- `nl.planon.cs.dam.ContainerSX`
- `nl.planon.cs.dam.ReportSX`
- `nl.planon.cs.dam.CalculateSX`



Configuring ContainerSX in Field Definer

The ContainerSX extension is configured in the **Data Aggregation Definition** business object:

On the **Extension** tab, add to the following extension:

- Extension = `nl.planon.cs.dam.ContainerSX`

- Event type = AI, AfterDBInsert
- Sequence of execution = 1

The content of the **Parameter** field contains all configurable parameters, necessary for the ContainerSX to work properly. The complete content must be:

Key	Value example / Description
Container.BO.Name	DataAggregationDefinition Description The name of the business object that holds the definition (container) records (current business object)
Data.BO.Name	DataAggregationData Description The name of the business object that holds the benchmark data level. This level has a 1-n relation with the definition (container) level.
BenchmarkEntity.BO.Name	Property Description The name of the business object that holds the records (entities) that must be benchmarked. Typically, this is the Property business object.
Data.BO.ContainerRef.Name	DataAggregationDefinitionRef Description The name of the field of the data level business object which holds the link to the definition (container) level business object.
Data.BO.Benchmark-EntityRef.Name	FreeString12 Description The name of the field of the data level business object that holds the link to the business object of the entities to benchmark. Typically, this is a link to the Property business object.
Data.BO.ParentRef.Name	ParentRef Description The name of the field of the data level business object that holds the link to the data

Key	Value example / Description
BenchmarkEntity.BO.- ParentRef.Name	level business object that in the hierarchy is the 'parent' of the current business object AlternativeParentRef Description The name of the field of the business object of the entities to benchmark which holds the link to a similar business object that is to be the parent in the hierarchical benchmark structure.
BenchmarkEntity.BO.- BenchmarkRef.Name	FK_ALTERNATIVE_PARENT_PROPERTY Description Same as previous, but here the database name of the field.
Benchmark.Picklist.Name	BENCHMARK Description Name of code/descriptive pick list in which all benchmarks that can be used are specified.
Benchmark.Code	IPD Description The code of the specific benchmark for which purpose the Data business object is configured (is part of Benchmark.Picklist.Name)
Create.Hierarchical.Structure	false or true Description When generating the data business objects, a hierarchical structure is created as defined by the parameters Data.BO.ParentRef.Name, BenchmarkEntity.BO.ParentRef.Name and BenchmarkEntity.BO.BenchmarkRef.Name. However, if this parameter is set to false , the hierarchical structure will not be created and the three previously mentioned parameters will be ignored.
Strict.ParentRef.Check	false or true Description If set to true (default value) then, when creating a hierarchical structure, all members of the hierarchy must have a link to the specified benchmark code. If that is not the

Key	Value example / Description
Error.Mandatory.- Parameters.Missing	<p>case, then the data business objects of this hierarchical family are not created.</p> <p>However, if this parameter is set to false, then the data business objects will be created if the corresponding benchmark entity has a link to the specified benchmark code. The hierarchical structure will only be created if both parent and child have this link to the specified benchmark code.</p> <p>Example</p> <p>In the hierarchy, if the grandparent and grandchild have a link to the specified benchmark code, but the parent does not, then data business objects will be created for both grandparent and grandchild, but in a flat list.</p> <p>Please provide parameter(s) :</p> <p>Description</p> <p>The default text of the error message that will appear if one or more parameters are missing.</p>
Error.Mandatory.- Parameter.Values.Missing	<p>Please provide value for parameter(s) :</p> <p>Description</p> <p>The default text of the error message that will appear if the values of one or more parameters are missing.</p>
Error.Invalid.Parameter.Value	<p>Invalid value for Parameter :</p> <p>Description</p> <p>The default text of the error message that will appear if one or more parameters are set to an invalid value.</p>

When the extension is configured on the business object with correct layouts, TSI and authorizations are configured, you can create a definition (container) record.

After inserting a new definition (container) record the ContainerSX will be executed. This SX will create the data level records based on the specified parameters. The records will be created in the specified hierarchical structure (unless the parameter **Create.Hierarchical.Structure** is set to **false**).

Example of Parameter field

```
Container.BO.Name=UsrAggregationDefinition2
```

```

Data.BO.Name=UsrAggregationData
BenchmarkEntity.BO.Name=Property
Data.BO.ContainerRef.Name=DataAggregationDefinitionRef
Data.BO.BenchmarkEntityRef.Name=FreeString12
Data.BO.ParentRef.Name=ParentRef
BenchmarkEntity.BO.ParentRef.Name=AlternativeParentRef
BenchmarkEntity.BO.BenchmarkRef.Name=FK_ALTERNATIVE_PARENT_PROPERTY
Benchmark.Picklist.Name=BENCHMARK
Benchmark.Code=IPD
Create.Hierarchical.Structure=true
Strict.ParentRef.Check=false
Error.Mandatory.Parameters.Missing=Please provide parameter(s):
Error.Mandatory.Parameter.Values.Missing=Please provide value for parameter(s):
Error.Invalid.Parameter.Value=Invalid value for Parameter:

```

Configuring ReportSX in Field Definer

The ReportSX extension is configured, for example, for the **Aggregation definitions** business object.

On the **Extension** tab add the following solution extension:

- Extension = *nl.planon.cs.dam.ReportSX*
- Event type = BU, BeforeDBUpdate
- Sequence of execution = 1

The content of the **Parameter** field contains all configurable parameters necessary for the ReportSX to work properly. The complete content must be:

Key	Value example / Info
Debug	y
	Info
	When this parameter is set to 'y' debug logging is switched on, which is useful for testing SQL statements while developing.
	When this parameter is left out, the default value of 'n' is taken.

Key	Value example / Info
aggregation.properties.bo.code	<p>aggregation.properties.bo.code = IPD corresponds with BO record: DataAggregationProperties.code = 'IPD'</p> <p>Info</p> <p>DAM definitions to fill the data and details fields (see ReportSX) can be given (with this key) in the Data aggregation properties business object.</p> <p>The code that is given here corresponds with the Code field value of the record in the Data aggregation properties business object.</p>
execute.on.status.change	<p>y or n</p> <p>Info</p> <p>If this parameter is set to 'y' the extension will only be executed if the definition (container) record is changed to a certain status.</p> <p>When this parameter is left out, the default value of 'n' is taken.</p> <p>If this parameter is set to 'n' the extension will be executed on every update.</p>

Example of Parameter field

```
debug=y
aggregation.properties.bo.code=Report SX DAM
execute.on.status.change=true
```

Configuring CalculateSX in Field Definer

The CalculateSX extensions can be configured for any Planon business object.

On the **Extension** tab, add the following solution extension:

- Extension = *nl.planon.cs.dam.CalculateSX*
- Event type = BU, BeforeDBUpdate
- Sequence of execution = 1

The content of the **Parameter** field contains all configurable parameters necessary for the CalculateSX to work properly. The complete content must be:

Key	Value example / Info
Debug	<p>y or n</p> <p>Info</p> <p>When this parameter is set to 'y' debug logging is switched on, which is useful for testing SQL statements while developing.</p> <p>When this parameter is left out, the default value of 'n' is taken.</p>
aggregation.properties.bo.code	<p>aggregation.properties.bo.code = IPD</p> <p>corresponds with BO record:</p> <p>DataAggregationProperties.code = 'IPD'</p> <p>Info</p> <p>DAM definitions to fill the data and details fields (see CalculateSX) can be given (with this key) in the Data aggregation properties business object.</p> <p>The code that is given here corresponds with the Code field value of the record in the Data aggregation properties business object.</p>
execute.on.status.change	<p>y or n</p> <p>Info</p> <p>If this parameter is set to 'y', this extension will only be executed if the definition (container) record is changed to a certain status.</p> <p>When this parameter is left out, the default value of 'n' is taken.</p> <p>If the CalculateSX is triggered by the update of a field that you just changed and this field is used in the SQL definitions, the old value and not the changed value is used in the SQL.</p>

Example of Parameter field:

```

debug=y

aggregation.properties.bo.code=Calculate SX DAM

execute.on.status.change=true

```

Data aggregation properties business object

As described earlier in the [ReportSX](#) and [CalculateSX](#) parameters, the definitions to fill the data and details fields can be provided in the **Data aggregation properties** business object.

On the **DAM TSI**, on the **Aggregation definitions** level, the **Aggregation properties** step is available.

The following fields are available:

Name	Type
Code	String(20) Description ID of the property file. In the SX parameter 'aggregation.properties.bo.code' this code is referenced.
Name	String(255) Description Optional description.
Properties	String extended Description Definitions to fill the data and details fields.
Comment	String extended Description Free available comment field.
Free fields	Standard set.

Configuring Data aggregation properties

The following sections describe how to configure the **Data aggregation properties** business object.

Aggregation properties BO for ReportSX and CalculateSX

The following keys can be configured for ReportSX and CalculateSX (using the **Data aggregation properties** BO).

General configuration

Key	Value example / Info	Extension
Container.BO.- Name	Container.BO.Name=- UsrDataAggregationDefinition Info <ul style="list-style-type: none">• Mandatory• System name of BO that holds the definition (container) records (level 1).	ReportSX
Data.BO.Name	Data.BO.Name=Usr- DataAggregationData Info <ul style="list-style-type: none">• Mandatory• System name of BO that holds the Data records (level 2).	ReportSX

Key	Value example / Info	Extension
Data.Container.- BO.Association	Data.Container.BO.- Association=DataAggregationData DataAggregationDefinitionRef Info <ul style="list-style-type: none"> • Mandatory • Association between Data BO and definition (container) BO. • This can be obtained from Field Definer. (Technical information tab, under Associations). 	ReportSX
State.Name.- Update	State.Name.Update=UsrUpdate Info <ul style="list-style-type: none"> • Mandatory • ReportSX: System name of statuses in which the definition (container) BO must be, for updating the Data BOs. • CalculateSX: System names of statuses for updating the Calculate BO. 	ReportSX/ CalculateSX
State.Name.- AfterUpdate	State.Name.AfterUpdate=UsrCheck Info <ul style="list-style-type: none"> • Mandatory • ReportSX: System name of status in which the definition 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
Data.BO.Modify- State.Name	<p>(container) BO must be set after updating the Data BOs.</p> <ul style="list-style-type: none"> CalculateSX: System name of status in which the Calculate BO must be set after updating it. <p>Data.BO.ModifyState.- Name=UsrModifyAll;</p> <p>Info</p> <ul style="list-style-type: none"> Mandatory System name of state in which the Data BO must be to have it updated. List of statuses must be divided by ; 	ReportSX
Log.<flag>	<p>Log.TRUE =FreeRemark2</p> <p>Info</p> <ul style="list-style-type: none"> ReportSX: System name of field of definition (container) BO in which the logging must be stored. CalculateSX: System name of field of Calculate BO in which the logging must be stored. Format Log.<FLAG>=<Comment- FieldName> 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
Log.Detail	<ul style="list-style-type: none"> • Possible values for flag TRUE and FALSE <p data-bbox="673 396 870 424">Log.Detail =high</p> <p data-bbox="673 443 727 470">Info</p> <ul style="list-style-type: none"> • Possible values: high, low • 'high' is the default value and will give the complete log along with the executed query statements. • 'low' will only log the value obtained from the query. 	ReportSX/ CalculateSX
Log.Overwrite	<p data-bbox="673 1024 911 1052">Log.Overwrite =true</p> <p data-bbox="673 1071 727 1098">Info</p> <ul style="list-style-type: none"> • Possible values: true, false • 'true' will overwrite the existing log. • 'false' is the default value and will append the log text to the existing log. 	ReportSX/ CalculateSX
LogOnData-Level.<flag>	<p data-bbox="673 1549 967 1606">LogOnDataLevel.TRUE- =FreeRemark2</p> <p data-bbox="673 1625 727 1652">Info</p> <ul style="list-style-type: none"> • ReportSX: System name of field of data BO in which the logging must be stored. 	ReportSX

Key	Value example / Info	Extension
LogOnData-Level.Overwrite	<ul style="list-style-type: none"> • Format LogOnDataLevel.- <FLAG>=<Comment- FieldName> • Possible values for flag TRUE and FALSE • When this parameter is set, the logging will be per data entity. If not set, the logging will be at definition level. <p>LogOnDataLevel.Overwrite=true</p> <p>Info</p> <ul style="list-style-type: none"> • Possible values: true, false • 'true' will overwrite the existing log on data level. • 'false' is the default value and will append the log text to the existing log on data level. 	ReportSX
Suppress.Confirmation	<p>Suppress.Confirmation= false</p> <p>Info</p> <ul style="list-style-type: none"> • Possible values: true, false • Confirmation to launch ReportSX or CalculateSX: • 'true' no confirmation 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
	<ul style="list-style-type: none"> 'false' (default value) confirmation needed by user 	
Container.BO.sourceID.Name	<p>Container.BO .sourceID.Name =FreeString1</p> <p>Info</p> <ul style="list-style-type: none"> If this parameter is given, the configured Container field is filled with the property file name (not the path) or code of the last executed properties file or BO. The Container field must be of type <i>String</i>. The Container BO can be 'DataAggregationDefinition' or any other configured BO. For CalculateSX, the container is the BO on which it is registered in Field Definer (see Configuring CalculateSX in Field Definer). 	ReportSX/ CalculateSX

Example using the Data aggregation properties BO

```
debug=y
aggregation.properties.bo.code=IPD
```

```
execute.on.status.change=true
```

Variables to use in SQL

You can define your own variables, based on Planon ProCenter definition (container) and data level BOs, to use in your SQL. The following types are supported:

- STRING
- INTEGER
- DECIMAL
- REFERENCE (referenced field)
- DATETIME
- SYSCODE (primary key field)
- CODESCODENAME (pick list code descriptive)
- CODESNAME (pick list descriptive)
- DATETIME_PROPERTY
- DATETIME_TRANSACTION
- BOOLEAN

You can define more than one variable. These variables can be used in the SQL for data definitions and in other variables beginning and ending with '&' character (see examples). These variables can be used as parameters in SQL for both data and details level and as parameters in other variable definitions.

There are 4 different kinds of variables you can define and use:

- Field (variable contains field value of definition (container) of data level BO)
- Constant (variable contains a constant value)
- SQL
- Expression

Key	Value example / Info	Extension
Variable.Field.- <VariableName>.- <TYPE>	Variable.Field.PropertyRef.- REFERENCE.INTEGER- =Data.BO.Name.Free- String12 (&PropertyRef&) Variable.Field.ContainerRef.- REFERENCE.INTEGER = Data.BO.Name.DataAggregation- DefinitionRef (&ContainerRef&) Variable.Field.StartDate.DATETIME	ReportSX/ CalculateSX

Key	Value example / Info	Extension
	= Container.BO.Name.BeginDate (&StartDate&)	
	<p>Info</p> <ul style="list-style-type: none"> • Variable to be populated.- Format: Variable.Field.- <VariableName>.- <TYPE>=<Field-Name> • ReportSX: The value should contain the BO name and the field name. For example : Data.BO.Name.- PropertyRef • CalculateSX: The value should only contain the field name. For example PropertyRef • If it is a reference field then the reference field type should be configured as shown in the following example : Variable.Field.- PropertyRef.- REFERENCE.- INTEGER • ReportSX: BO can only be Data.BO.Name (data level fields) or Container.BO.Name. (definition 	

Key	Value example / Info	Extension
Variable.Constant.- <VariableName>.- <TYPE>	<p>(container level)</p> <ul style="list-style-type: none"> CalculateSX: only fields of the Calculate BO can be defined. <p>Variable.Constant.HighVatRate.- DECIMAL=1.21 (&HighVatRate&) Variable.Constant.- UnitToLetOut.STRING- =UsrUnitToLetOut (&UnitToLetOut&) Variable.Constant.- RefDateSpec.DATE=20/01/2014 (&RefDateSpec&)</p> <p>Info</p> <ul style="list-style-type: none"> Variable to be populated. Format: Variable.Constant.- <VariableName>.- <TYPE>.- =<constant value> Only types STRING, INTEGER, DECIMAL, DATE and DATETIME are allowed for this type of variable. No quotes needed for STRING, DATE and DATETIME values Decimal separator for DECIMAL type always . (dot) 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
Variable.SQL.- <VariableName>.- <TYPE>	<ul style="list-style-type: none"> • Format value of DATE type: "dd/MM/yyyy" • Format value of DATETIME type: "dd/MM/yyyy hh:mm:ss" <p>Variable.SQL.WeeklyRateCompany-Car.DECIMAL= { Select WEEK From TRFGRP Where CODE = 'CC' } (&WeeklyRateCompanyCar&)</p> <p>Info</p> <ul style="list-style-type: none"> • Variable to be populated. Format: Variable.SQL.- <VariableName>.- <TYPE>=<- SQL-select statement> • Only types STRING, INTEGER, DECIMAL, DATE and DATETIME are allowed for this type of variable. • SQL string can be given as one line with no carriage returns or can be divided over more lines. In the last method the query must be started with { character and ended with } 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
Variable.Expression.<Variable-Name>;<TYPE>	Variable.Expression.- WeeklyRateCompCar- InclVat.DECIMAL=& WeeklyRate- CompanyCar& * & HighVatRate& (&WeeklyRateCompCarInclVat&) Variable.Expression.- NrWPTotal.DECIMAL=&NrFlex& + &NrFixed& (&NrWPTotal&)	ReportSX/ CalculateSX
	Info <ul style="list-style-type: none"> • Variable to be populated. Format: Variable.Expression.- <VariableName>. <TYPE>=- <expression> • Add/Subtract support: STRING, INTEGER, DECIMAL • Multiply/Divide support: INTEGER, DECIMAL • Only simple expressions with two operands and one operator (+, -, * or /) are allowed here. 	

SQL for Data level

The following sections provide an overview of the fields and SQL configuration for the Data level.

Data level

Based on SQL queries, specific fields on the Data level can be filled.


Key	Value example / Info	Extension
Field.<FieldName>.<TYPE>	<p>Field.FreeString41.STRING= select Free30 from OBJALG where syscode = &PropertyRef& or Field.FreeString41.STRING= { select Free30 from OBJALG where syscode = &PropertyRef& } Field.FreeDecimal1.DECIMAL= select m2bvo from OBJALG where syscode = &PropertyRef& or Field.FreeDecimal1.DECIMAL= { select m2bvo from OBJALG where syscode = &PropertyRef& } Field.FreeDecimal10.- DECIMAL; Field.FreeDecimal11.- DECIMAL =select Kadoppvl, Free25 from OBJALG where syscode = &PropertyRef& or Field.FreeDecimal10.DECIMAL; Field.FreeDecimal11.DECIMAL = { select Kadoppvl, Free25 from OBJALG where syscode = &PropertyRef& }</p> <p>Info</p> <ul style="list-style-type: none"> • Fields to be populated. Format: Field.- <FieldName>. <TYPE>- =<SQL-select statement> <p>Available types are:</p> <ul style="list-style-type: none"> ◦ STRING ◦ INTEGER ◦ DECIMAL ◦ REFERENCE.INTEGER and REFERENCE.STRING ◦ DATE / DATETIME ◦ CODESCODENAME ◦ CODESNAME ◦ DATETIME_PROPERTY ◦ DATETIME_TRANSACTION <ul style="list-style-type: none"> • If more than one field is to be 	ReportSX/ CalculateSX

Key	Value example / Info	Extension
	<p>updated, then separate them with a semicolon (;) as shown in the last example.</p> <ul style="list-style-type: none"> All the variables in the query (for example: &PropertyRef&) should be defined. All the variables should start and end with an "&" symbol SQL string can be given as one line with no carriage returns or can be divided over more lines. In the last method the query must start with { character and end with } 	

Details level

Based on SQL queries, Details level records can be filled. Consequently, these queries can result in more records with more fields. You can define more BOs to process as details level. These BOs have your own associated keys in the **Aggregation properties** BO, based on their corresponding BO name.

Key	Value example / Info	Extension
Detail.BO.Name	<p>Detail.BO.Name=- UsrDataAggregationDetail1; usrDataAggregationDetail2</p> <p>Info</p> <ul style="list-style-type: none"> System name of BO. 	ReportSX

 If more than one details BO has to be created and updated, then separate

Key	Value example / Info	Extension
<BO name>.Data.- BO.Association	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p>them with a semicolon (;) as shown in the example.</p> </div> <p>UsrDataAggregationDetail.Data.BO.- Association= DataAggregationDetail DataAggregationDataRef</p> <p>Info</p> <ul style="list-style-type: none"> • Mandatory for each system BO that is specified by the previous parameter • Association between Details BO and Data BO. • This can be obtained from Field Definer (Technical information tab, under Associations). 	ReportSX
<BO name>. ProcessDetail- RecordsForNon- UpdatableData- Records	<p>UsrDataAggregationDetail.Process- DetailRecordsForNonUpdatableData- Records=n</p> <p>Info</p> <ul style="list-style-type: none"> • Possible values: y,n • If value is y: also for the Data level records that are not in a status for update, the Details records will be deleted and inserted again. • If value is n: for the Data level records that are not in a status for update, 	ReportSX

Key	Value example / Info	Extension
<BO name>.SQL.- Statement.<serial number>	<p>the Details records will not be deleted nor updated or inserted.</p> <p>UsrDataAggregationDetail.SQL.- Statement.1= select CODE as PropertyCode, NAAM as PropertyName, M2BVO as GrossArea from OBJALG where SYSCODE = &PropertyRef&</p> <p>or</p> <p>UsrDataAggregationDetail.SQL.- Statement.1=</p> <pre>{ select CODE as PropertyCode, NAAM as PropertyName, M2BVO as GrossArea from OBJALG where SYSCODE = &PropertyRef& }</pre>	ReportSX
	<p>Info</p> <ul style="list-style-type: none"> • SQL select query to create specific details BO and fill its fields. • You can specify more than one SQL statement for one details BO. Each identified by its own unique serial number. 	
	<p>Example</p> <p>UsrDataAggregationDetail.SQL.- Statement.1= SELECT ...</p> <p>UsrDataAggregationDetail.SQL.- Statement.2= SELECT ...</p> <p>Records from both queries will be added,</p>	

Key	Value example / Info	Extension
<p data-bbox="748 247 1016 310">field assignment are shared, see next key.</p> <ul data-bbox="703 331 951 726" style="list-style-type: none"> <li data-bbox="703 331 951 726">• SQL string can be given as one line without carriage returns or can be divided over more lines. In the last method, the query must start with { character and end with } <p data-bbox="316 762 542 884"><BO name>.SQL.- Field.<sqlfield name>. <type>- =<BO field name></p>	<p data-bbox="670 247 1127 310">field assignment are shared, see next key.</p> <ul data-bbox="703 331 951 726" style="list-style-type: none"> <li data-bbox="703 331 951 726">• SQL string can be given as one line without carriage returns or can be divided over more lines. In the last method, the query must start with { character and end with } <p data-bbox="670 762 1127 982">UsrDataAggregationDetail.SQL.Field.- PropertyCode.STRING=FreeString1 UsrDataAggregationDetail is BO system name of details BO, PropertyCode is SQL select field, FreeString1 is system name of BO field that has to be filled.</p> <p data-bbox="670 1003 1127 1157">UsrDataAggregationDetail.SQL.- Field.PropertyName.- STRING=FreeString2 UsrDataAggregationDetail.SQL.Field.Gross- Area.DECIMAL=FreeDecimal3</p>	<p data-bbox="1247 247 1386 793">ReportSX</p>
	<p data-bbox="670 1178 727 1205">Info</p> <ul data-bbox="703 1226 951 1885" style="list-style-type: none"> <li data-bbox="703 1226 951 1352">• Fields of details level to fill, based on the SQL. <li data-bbox="703 1373 951 1730">• The fields can be of all the supported types. The types of the selected fields must match with the Planon ProCenter BO fields that are linked to it. <li data-bbox="703 1751 951 1885">• The SQL must select the configured fields, you can 	

Key	Value example / Info	Extension
	define as many fields as you need. <ul style="list-style-type: none"> The defined SQL parameters can be used in the same way as for the Data level. 	

Aggregation of fields

The data level can have a hierarchical structure. With the following keys you can define to add the values of child records to their parent. The summary is processed bottom up, from deepest to root level. So the values of grand children are added to their parent, all the values of the parents are added to the grandparents and so on.

Key	Value example / Info	Extension
SUM.DECIMAL	SUM.DECIMAL=Free-Decimal1,FreeDecimal10, Free-Decimal11 Info <ul style="list-style-type: none"> Aggregation of fields. Specify DECIMAL fields. The aggregation is performed for all the listed fields in the value part. These fields need not necessarily be populated by this SX. If you do not want to execute aggregate, 	ReportSX

Key	Value example / Info	Extension
SUM.DECIMAL.OVERWRITE	<p data-bbox="769 249 889 344">the value should be empty.</p> <p data-bbox="688 380 1122 474">SUM.DECIMAL.OVERWRITE=Free-Decimal1,FreeDecimal10, Free-Decimal11</p> <p data-bbox="688 491 740 518">Info</p> <p data-bbox="688 537 1081 659">Same as for SUM.DECIMAL, but the value of the parent record is not taken into account. Its value is the sum of all child values.</p> <p data-bbox="688 680 802 707">Example</p> <p data-bbox="688 726 1081 915">Value child 1 is 10, value child 2 is 11, value of the parent is 12. After aggregation, the value will be 33 if the OVERWRITE property is not specified or 21 if it is specified.</p>	ReportSX
SUM.INTEGER	<p data-bbox="688 953 1062 1016">SUM.INTEGER=FreeInteger8,-FreeInteger10</p> <p data-bbox="688 1033 740 1060">Info</p> <ul data-bbox="721 1079 922 1890" style="list-style-type: none"> <li data-bbox="721 1079 922 1247">• Aggregation of fields. Specify INTEGER fields. <li data-bbox="721 1266 922 1493">• The aggregation is performed for all the listed fields in the value part. <li data-bbox="721 1512 922 1709">• These fields need not necessarily be populated by this SX. <li data-bbox="721 1728 922 1890">• If you do not want to execute aggregate, the value 	ReportSX

Key	Value example / Info	Extension
SUM.INTEGER.OVERWRITE	<p data-bbox="769 247 891 310">should be empty.</p> <p data-bbox="688 348 1122 436">SUM.INTEGER.OVERWRITE=Free-Decimal1,FreeDecimal10, Free-Decimal11</p> <p data-bbox="688 457 1081 625">Info Same as for SUM.INTEGER, but the value of the parent record is not taken into account. Its value is the sum of all child values.</p>	ReportSX

Scheduling of ReportSX or CalculateSX

Scheduling

It is possible to run ReportSX and CalculateSX manually by changing the status of the definition (container) level record. However, it is also possible to schedule this action. Periodically an update run will be launched automatically.

Scheduling process

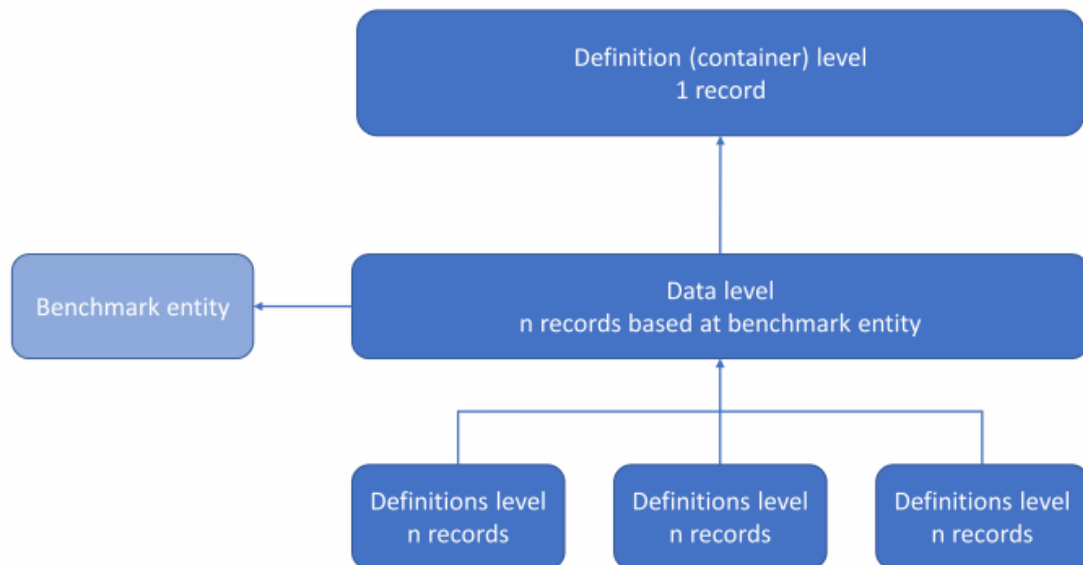
This scheduling task can be configured by using the built in scheduler. In fact, a status change of the specific definition (container) record must be performed. This function is available for the dedicated definition (container) BO (Data aggregation definition).



For more information refer to [Scheduler](#) (System Settings).

Working with Data Aggregation Manager

Data Aggregation Manager is primarily used for setting up a benchmark structure and filling this structure with relevant management data. The final structure can have 3 levels of data:



- **Definition (container) level**
Business object to define the benchmark with associated date fields etc.
- **Data level**
Business object linked to the definition (container) level with records of aggregated data. There can be more data level records linked to the same definition (container). It can have a hierarchical structure.
The data level records are based on the so called 'benchmark entity' business object. The members of the benchmark are defined at the benchmark entity business object (e.g. Property). Only for the members, a **Data level record** is added by **DAM**.
- **Details level**
Business objects linked to the data level with records of aggregated data. There can be more details level business objects for the same data level business object. Each details level business object can have more records.

ContainerSX

This is an extra component available to fill in the aggregated data, but not in the specific benchmark structure.

The definition (container) record must be filled in manually. Some of the fields are mandatory.

After saving the new definition (container) level record, the following step is processed:

Data level structure is created based on the benchmark entity business object (see parameters in Field Definer). Only the members of the specified benchmark are added. For each member, a record is added linked to the original benchmark entity record. At this data level, a large set of empty fields is available. These can be filled by the **ReportSX** process. Typically, the Property business object is used as benchmark entity business object, so it is possible to process a benchmark for each property (building). At the benchmark entity business object, a field is configured to filter the members.

ReportSX

If the status of the definition (container) record for the specific benchmark is changed to the configured statuses, the ReportSX is launched. After launch, the following steps are processed:

- Data is filled in configured fields at data level.

Fields are filled based on SQL queries in field.Properties in the **Aggregation properties** business object.

- Sum fields are processed.

If certain fields are configured for aggregation, the values of the child fields are added to the parents. If a certain field is not configured to be updated, it will not be changed in an update situation (record already exists).

- Clean up existing details level business object records.

Before the details level records and fields are filled, it starts with deleting the already existing records. There is a configuration that this is only performed for the details level records that are not linked to a data level record that is flagged for not updating.

- Details level business object records are added and fields are filled with data.

The details level records are added and the fields are filled as configured.

- The status of the definition (container) level business object record is updated.

After a successful calculation of data the status of the definition (container) level business object record will be changed. If the processing by ReportSX is not successful, an error pop-up will be displayed. Finally, logging of the run is performed in the **Comment** field of the definition (container) level record.

CalculateSX

The CalculateSX is a simple version of the ReportSX and not specially intended for a (benchmark) container construction. The CalculateSX can be configured for every BO in order to aggregate data for it.

If the status of the business object is changed to the configured statuses, the CalculateSX is launched. After launch, the following steps are processed:

- Data is filled in configured fields of the business object.
Fields are filled based on SQL queries in field.Properties in the **Aggregation properties** business object.
- The status of the business object is updated.
After a successful calculation of data, the status of the business object record will be changed. If the processing by CalculateSX is not successful, an error pop-up will be displayed. Finally, logging of the run is performed in the **Comment** field of the business object record.



Authorization in DAM

Since the data aggregation of the **DAM** is based on SQL, it is possible to view data that is usually not visible for certain Planon ProCenter users. So, only users who have appropriate access to the set of business objects used should be allowed to work with the definition (container), data and details business objects.

DAM implementation steps

To use the DAM (ContainerSX and ReportSX) you have to follow the steps given below:

Step	Description
Register extensions	Register DAM extensions in SX Configuration (see Registering extensions).
Prepare business objects	<ol style="list-style-type: none">1. Choose business object for the following levels of data:<ul style="list-style-type: none">◦ Definition (container) level. For example, Data aggregation definition.◦ Data level. For example, Data aggregation data.2. Prepare which fields are to be filled with what data by DAM.

Step	Description
	<ol style="list-style-type: none"> 3. Benchmark Entity BOs. Choose a benchmark entity business object (e.g. Properties) to use as source BO for the data level (see Configuring ContainerSX in Field Definer). 4. Details level (can be more than one). E.g. Data aggregation details (make user defined of it). 5. For each details level BO, you need to prepare which field and what data should be filled by DAM.
Prepare layouts and TSI	After preparing the business objects, in Layouts prepare the corresponding layouts to make sure the fields to be filled are available.
Configure tool	<p>After you prepared all the BOs, layouts and TSI, you can configure the DAM:</p> <ol style="list-style-type: none"> 1. Add extensions to the definition (container) level business object in Field Definer (see Configuring ContainerSX in Field Definer). 2. Configure parameters.
Aggregation properties BO	In the Property field, include all SQL related settings. Based on the SQL, the aggregated data will be created.
	<div style="border: 1px solid #00a0e3; padding: 5px;"> <p> Creating SQL queries is a specialist task that requires appropriate skills. This task should be done by a Planon ICT specialist.</p> <p> For more information about the data structure of the database, see the Planon data dictionary, an HTML page that contains detailed information of the Planon database and which is updated each time the data structure is changed.</p> </div>
Insert definition (container) record	Add a definition (container) record manually; DAM automatically constructs the data level structure based on the configured benchmark entity BO after saving.
Change status of definition (container) record	If the status of a definition (container) record is changed to one of the configured statuses, the fields at the data level will be filled based on the SQL definitions. Also, the details level records will be created and filled with data.

Step	Description
	<p>If the data and details level record are already filled, you can update them to change to one of the configured statuses.</p>
	<p>Updating is available for the status of the data level record. If they are in the right status, to update the configured data level, fields are overwritten and the details level records are deleted and added again.</p>
	<p>If the data level records are not in the right status, the existing fields and details level record will be unchanged (is configurable for details level).</p>

Field descriptions

The following section(s) describe(s) the fields, their purpose and meaning.

Aggregation data - fields

The following fields are available on **Aggregation data** level.

PnName	Translated name
ParentRef	Parent level
HierarchyCode	Hierarchy code
DataAggregationDefinitionRef	Aggregation definition
RefBOStateSystem	System status
RefBOStateUserDefined	User-defined status
RefBODefinitionUserDefined	User-defined type
SysChangeDateTime	Modification date-time
SysInsertDateTime	Insertion date-time
SysChangeAccountRef	Modified by
SysAccountRef	Inserted by
Money1 -50	Money field 1 - 50
Area1-15	Area 1-15
FreeInteger1-20	Free reference field 1-20
FreeDecimal1-100	Free numerical field 1-100
FreeDateTime1-20	Free date-time field 1-20
FreeRemark01-02	Free remark field 1-2



In Field Definer, it is possible to include a user defined system name for *free fields*: Area, Money, FreeInteger, FreeDecimal, FreeDateTime, FreeRemark. This will allow you to provide a name that fits its purpose and use that for filtering in your data lake, for example.

Aggregation details - fields

The following fields are available on **Aggregation details** level.

PnName	Translated name
DataAggregationDataRef	Aggregation data
BOType	Business object type
Code	Code
SysAccountRef	Inserted by
SysInsertDateTime	Insertion date-time
SysChangeDateTime	Modification date-time
SysChangeAccountRef	Modified by
Name	Name
SysDataSectionRef	Property set code
Syscode	System code
RefBOStateSystem	System status
SysUpdateCount	Update count
RefBOStateUserDefined	User-defined status
RefBODefinitionUserDefined	User-defined type
Money1 -50	Money field 1 - 50
Area1-15	Area 1-15
FreeString1-60	Free field 1-60
FreeInteger1-20	Free reference field 1-20
FreeDecimal1-100	Free numerical field 1-100
FreeDateTime1-20	Free date-time field 1-20
FreeRemark01-02	Free remark field 1-2



In Field Definer, it is possible to include a user defined system name for *free fields*: Area, Money, FreeInteger, FreeDecimal, FreeDateTime, FreeRemark. This will allow you to provide a name that fits its purpose and use that for filtering in your data lake, for example.

Index

A

- Aggregation data - fields 41
- Aggregation details - fields 42
- Authorization 38

C

- CalculateSX 38
- CalculateSX: Configuring 13
- CalculateSX: Field Definer 13
- ContainerSX 37
- ContainerSX: configure in Field Definer 8

D

- DAM 6
- DAM: implementation steps 38
- Data Aggregation Manager 6
- Data aggregation properties configuration 16
- Data level 26
- Data level: field aggregation 32
- Data level: fields 26
- Data level: SQL 26
- Details level 28

F

- Field aggregation 32
- Field descriptions 41

G

- General configuration 16

R

- Register extensions 8
- ReportSX 37
- ReportSX or CalculateSX: schedule 35
- ReportSX: Configuring 12
- ReportSX: Field Definer 12

S

- Scheduling process 35, 35
- SQL variables 22
- SX Configuration 8

W

- Working with DAM 36