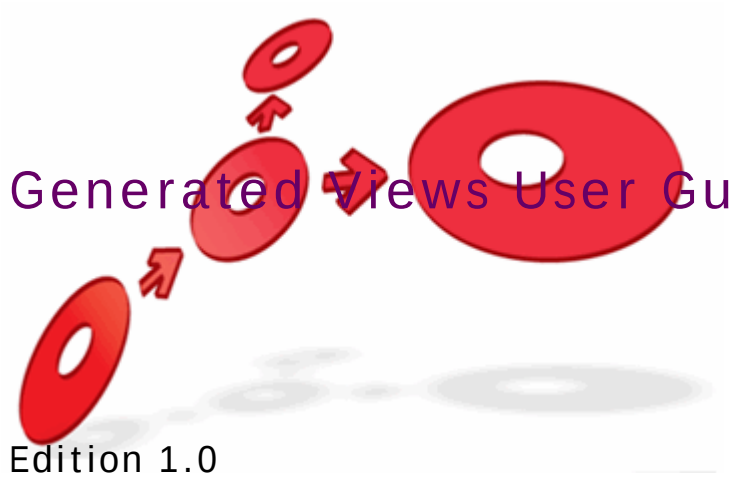
Portrait
Foundation



Generated Views User Guide

Edition 1.0

31 July 2017

Pitney Bowes
Software

Foundation
Generated Views User Guide

Acknowledgement of trademarks

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK

Portrait Software
The Smith Centre
The Fairmile
Henley-on-Thames
Oxfordshire, RG9 6AB, UK

Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778
Fax: +44 (0)1491 416601

America

Portrait Software
125 Summer Street
16th Floor
Boston, MA 02110
USA

Email: support@portraitsoftware.com
Tel: +1 617 457 5200
Fax: +1 617 457 5299

Norway

Portrait Software
Portrait Million Handshakes AS
Maridalsveien. 87
0461 Oslo
Norway

Email: support@portraitsoftware.com
Tel: +47 22 38 91 00
Fax: +47 23 40 94 99

About this document

Purpose of document

This document describes how to use the Generated views feature introduced in Portrait Foundation 5.0 U4 that was initially added for the purpose of data paging.

Intended audience

Developers and configurers who need to access CBE data in a normalised state from the Portrait Foundation operational database rather than using the Datamart.

Software release

Portrait Foundation 5.4 or later.

Contents

1 Introduction

1.1 Document Purpose

This document describes the database extensions provided to produce generated databse views and the installation steps for adding them to a Portrait Foundation implementation.

1.2 Rationale

The Portrait Foundation database is designed to accommodate the storage of any configured business entity without having to make any underlying database schema changes. This generic design enables rapid solution development but makes accessing the raw physical tables directly more difficult as a translation from the logical to the physical model is required using the deployed metadata.

The following entity types in a Portrait Foundation solution have configurable attributes that cannot be easily accessed from the operational database via the physical tables:

- Parties
- Repeating Attributes
- Contracts
- Product Definitions
- Engagement Actions
- Significant Events
- Case
- Milestones
- Workflow Tasks
- Complex Data Objects (CDOs)

These items are accessed in a Foundation implementation using the built-in Data Access Transactions (DATs) or by views that have been explicitly configured in the Configuration Suite and generated.

Reporting and data extraction from a Foundation database is typically performed by creating a DataMart from the Operational database. The DataMart database is structured in a more traditional way with a database table for each entity type. This has the advantage of making writing queries for reports and data extraction easier. However, the downside is that population of the DataMart requires all data to be copied from the Operational database. Updates to the DataMart are incremental and only data that has been added or modified is copied over. The exception is where entity definitions have been modified and deployed – this requires a full DataMart rebuild (as the schema needs to be updated) and can be a time consuming process if the data volumes are high.

The solution offered through the auto-generated database views, outlined in this document, provides access to the data in the Operational database through views that present the data in its logical form. This removes the need for a DataMart build when the data can be extracted or queried in-situ. This ultimately saves time in making the data available and reduces the storage requirements of the system.

1.3 Generating Database Views

Database views are generated using the stored procedures deployed as part of database upgrade.

Call each of the stored procedures described in the following section to generate the Entity views required by your implementation.

Calls to the stored procedures to refresh the views are performed by:

Manually running the procedures when updates are required

Running SQL Agent Job named like

amc_utl_update_generatedviews_<T_SQL_dbname>

A trigger is fired on the Portrait Foundation deployment table to force view updates whenever new configuration is deployed.

The following sections describe each of the stored procedures and views in more detail.

1.4 Database Extensions

The Generated Views extensions are made up of the following views and stored procedures:

Name	Type	Description
v_utl_cbe_definitions	View	A view on the CBE metadata.
v_engagement_action	View	A view that presents the engagement history and engagement actions in the same de-normalised format as the DataMart.
v_cdo_entities	View	View that transforms CDO xml to columns.
p_amc_utl_gen_entity_views	Stored Procedure	For generating CBE views
p_amc_utl_gen_refdata_views	Stord Proedure	For generating reference data views
p_ amc_utl_gen_task_views	Stored Procedure	For generating workflow task views
p_ amc_utl_gen_cdo_views	Stored Procedure	For generating CDO views
p_ amc_utl_gen_search_views	Stored Procedure	For generating CBE search attribute views

The views v_engagement_action and v_cdo_entities can be accessed directly as they do not vary according to entity type.

The stored procedures are used to generate views from the system metadata which can then be accessed to provide entity specific attributes. The stored procedures only need to be invoked when the generated views need to be refreshed, for example after a deployment has been made that changes one or more entity definitions.

1.5 Views

1.5.1 v_amc_utl_cbe_definitions

View that returns the metadata that describes each of the Configurable Business Entities (CBEs).

This view is used internally by the stored procedures that generate CBE views. It can be used to query the CBE metadata.

Returns a row for each attribute of a CBE.

Column name	Data type	Description
entity_id	int	Unique CBE object Id. Unique Id for each entity type.
cbe_definition_id	int	Unique CBE definition Id. Unique for each attribute.
cbe_rdg	int	The RDG type of the entity.
cbe_rdi	int	The RDI type of the entity.
level_name	varchar(50)	Entity name in the hierarchy.
level_order	int	Indicator of level in hierarchy. 0 = leaf.
entity_name	varchar(50)	The entity system name.
qualifier	varchar(31)	Text value for level order – used in joins
attr_name	varchar(50)	System name of attribute
qualified_attr_name	varchar(82)	Attribute name combined with qualifier.
data_type	varchar(50)	System name of data type
datatype_column	varchar(30)	The physical column name
ridrdg	int	If the attribute is of type Reference, the reference data group Id.
is_search_criteria	int	Flag indicating that the attribute is searchable
is_unqiue	int	Flag indicating that the attribute is unique

1.5.2 v_engagement_action

This view presents the engagement history and engagement actions in the same de-normalised format as the DataMart. In most cases, the columns map directly to the columns on the referenced tables.

Column name	Data type	Description
engagement_action_id	bigint	Unique key of each engagement action
eng_action_type_rdg	int	RDG of engagement action type
eng_action_type_rdi	int	RDI of engagement action type
eng_action_created_when	datetime	Created_when date of engagement action
contract_id	bigint	Related contract_id (if applicable)
product_rdg	int	Related product rdg (if applicable)
product_rdi	int	Related product rdg (if applicable)
case_id	bigint	Case Id (if applicable)
action_note	varchar(2000)	Engagement action notes
eng_action_start_date_time	datetime	Engagement action start datetime
eng_action_end_date_time	datetime	Engagement action end datetime
eng_action_duration	int	Difference in seconds between start and end datetimes
engagement_id	bigint	Unique key of the Engagement
eng_type_rdg	int	RDG of Engagement type
eng_type_rdi	int	RDI of Engagement type
eng_start_date_time	datetime	Engagement start datetime
eng_created_when	datetime	Engagement creation datetime
eng_end_date_time	datetime	Engagement start datetime
eng_duration	int	Difference in seconds between start and end datetimes
engagement_note	varchar(2000)	Engagement notes
eng_outcome_rdg	int	RDG of Engagement Outcome
eng_outcome_rdi	int	RDI of Engagement Outcome

1.5.3 v_cdo_entities

The CDO Entities view returns multiple rows for each CDO record in the *amc_cdo_instance* table. Each row represents an Element in the CDO XML instance data. The view must be ordered by cdo_instance_id to group all elements of a single CDO record together.

Column name	Data type	Description
cdo_instance_id	bigint	Unique ID of CDO Instance. Multiple rows with the same instance id can be returned in the rowset.
CDODefinition	varchar(50)	The CDO definition type name.
element	varchar(255)	The name of the Element in the XML. Values are typically: • CompositeDataObject • CollectionItem • CompositeDataObjectElement • DataObjectElement
type	varchar(255)	The value stored in the 'Type' attribute or NULL if not included. The Type attribute is only populated on the root node of the CDO XML and is the same as the CDODefinition type name.
version	varchar(255)	The value stored in the 'Version' attribute or NULL if not included. Typically = 0.
ID	bigint	The value stored in the 'ID' attribute or NULL if not included. This is the ID of the persisted entity – e.g. the party_id, contract_id etc or if the object is a CDO, the CDO Id.
encryptedID	varchar(50)	The value stored in the 'EncryptedID' attribute or NULL if not included. Encrypted value of the ID.
systemName	varchar(255)	The value stored in the 'SystemName' attribute or NULL if not included. This is typically the property name of the attribute in the CDO definition.
objectType	varchar(255)	The value stored in the 'ObjectType' attribute or NULL if not included. Values are: • DataObject • DataObjectCollection
compositeDataObject	varchar(255)	The value stored in the 'CompositeDataObject' attribute or NULL if not included. This is the type of the nested CDO instance.
dataObjectCategory	varchar(255)	The value stored in the 'Type' attribute or NULL if not included. The Category of data object e.g. Party, ProductDefinition, RepeatingAttribute.
dataObjectType	varchar(255)	The value stored in the 'Type' attribute or NULL if not included. The Data object type, e.g. Customer.
parentSystemName	varchar(255)	The value stored in the 'Type' attribute or NULL if not included. The System Name of the parent element.

1.6 Stored Procedures

1.6.1 p_amc_utl_gen_entity_views

This stored procedure is used to generate views for the following entity types: Parties, Repeating Attributes, Contracts, Product Definitions, Engagement Actions, Milestones, Case and Significant Events.

Syntax

```
p_utl_gen_entity_views
[   [ @p_drop_only = ] 'drop_only' ]
[ , [ @p_include_encrypted_pk = ] 'include_encrypted_pk' ]
[ , [ @p_cbe_type = ] 'cbe_type' ]
```

Arguments

[@p_drop_only =] '*drop_only*'

This is a flag to indicate that the views should be dropped or created. '*drop_only*' is BIT. 1 = Drop the views and do not create them, 0 indicates that the views will be dropped and re-created. The DEFAULT value is 0.

[@p_include_encrypted_pk =] '*include_encrypted_pk*'

This is a flag to indicate that the generated views should include a column for the encrypted primary key Id. '*include_encrypted_pk'* is BIT. The DEFAULT value is 0 (do not created encrypted primary key column).

[@p_cbe_type =] '*cbe_type*'

This is the type of cbe to generate views for. The default value is NULL which means that views for all of the entity types will be created. If specified it can be one of the following values:

Value	Description
1	Party
2	Repeating Attribute
3	Contract
20	Engagement Action
39	Significant Event
90	Milestone
91	Case
96	Product Definition

Return Code Values

0 (success) or 1 (failure)

Remarks

Views are created for the entity type specified or all entities.

The views are named with the following syntax:

v_<entity identifier>_<system_name>

where *entity identifier* is a value specific to each entity and follows the naming convention used in the DataMart table names. The values are:

p Party
ra Repeating Attribute

c	Contract
ea	Engagement Action
ev	Significant Event
mi	Milestone
ca	Case
pd	Product Definition

and *system_name* is the system name of the entity.

For example:

v_p_Consumer - Party = Consumer

v_c_BankAccount - Contract = BankAccount

v_ra_Address - Repeating Attribute = Address

The columns returned in each view included the following fixed and variable items:

<entity>_id	- Variable – Always provided but the name varies by entity
<entity>_id_encrypted	- Optional (based on flag used when generating the view)
< all configured columns >	- Entity specific. Includes all inherited attributes.
CREATED_WHEN	- Fixed
CREATED_BY	- Fixed
UPDATED_WHEN	- Fixed
UPDATED_BY	- Fixed
UPDATE_COUNT	- Fixed
EFFECTIVE_FROM	- All types except Milestone
EFFECTIVE_TO	- All types except Milestone

Repeating attribute views also include the following columns:

- PARTY_ID
- RELATED_PARTY_ID
- PARENT_REPEATING_ATTRIBUTE_ID

Reference data type attributes are provided as system name and display name values. The Attribute name is used as the column name containing the system name value and the Attribute name with '*_display*' appended contains the display name.

Examples

```
-- Generate all entity views
p_utl_gen_entity_views

-- Drop all entity views
p_utl_gen_entity_views 1
-- or
p_utl_gen_entity_views @p_drop_only = 1

-- Create all entities and include encrypted PK
p_utl_gen_entity_views  0, 1
-- or
p_utl_gen_entity_views  @p_include_encrypted_pk = 1
```

```
-- Create just Repeating Attribute views and include encrypted PK
p_utl_gen_entity_views  0, 1, 2
-- or
p_utl_gen_entity_views  @p_include_encrypted_pk = 1 , @p_cbe_type = 2
```

1.6.2 p_amc_utl_gen_refdata_views

This stored procedure is used to generate views for all reference data groups. The resultant generated views return one row for each reference data item and include all attributes.

Syntax

```
p_utl_gen_refdata_views
[   [ @p_drop_only = ] `drop_only' ]
```

Arguments

[@p_drop_only =] `*drop_only*'

This is a flag to indicate that the views should be dropped or created. `*drop_only*' is BIT. 1 = Drop the views and do not create them, 0 indicates that the views will be dropped and re-created. The DEFAULT value is 0.

Return Code Values

0 (success) or 1 (failure)

Remarks

Views are created with the following naming convention:

v_rd_<system_name>

where *system_name* is the system name of the reference data group.

For example:

v_rd_Titles = 'Titles' reference data group

Examples

-- Generate all reference data group views

p_utl_gen_refdata_views

-- Drop all entity views

p_utl_gen_refdata_views 1

-- or

p_utl_gen_refdata_views @p_drop_only = 1

1.6.3 p_amc_utl_gen_task_views

This stored procedure is used to generate views for the all workflow task types.

Syntax

```
p_utl_gen_task_views
[   [ @p_drop_only = ] 'drop_only' ]
[ , [ @p_include_encrypted_pk = ] 'include_encrypted_pk' ]
```

Arguments

[@p_drop_only =] '*drop_only*'

This is a flag to indicate that the views should be dropped or created. '*drop_only*' is BIT. 1 = Drop the views and do not create them, 0 indicates that the views will be dropped and re-created. The DEFAULT value is 0.

[@p_include_encrypted_pk =] '*include_encrypted_pk*'

This is a flag to indicate that the generated views should include a column for the encrypted primary key (task_instance_id). '*include_encrypted_pk*' is BIT. The DEFAULT value is 0 (do not created encrypted primary key column).

Return Code Values

0 (success) or 1 (failure)

Remarks

The views are named with the following syntax:

v_te_<system_name>

where *system_name* is the system name of the task type.

For example:

v_te_CallBack	- All 'Call Back' task types
v_te_ProcessApplication	- All 'Process Application' task types

The columns returned in each view included the following fixed and variable items:

task_instance_id	- Fixed
task_instance_id_encrypted	- Optional (based on flag used when generating the view)
< all configured columns >	- Task Definition specific
CREATED_WHEN	- Fixed
CREATED_BY	- Fixed
UPDATED_WHEN	- Fixed
UPDATED_BY	- Fixed

Reference data type attributes are provided as system name and display name values. The Attribute name is used as the column name containing the system name value and the Attribute name with '*_display*' appended contains the display name.

Examples

-- Generate all task views

p_utl_gen_task_views

-- Drop all task views

```
p_utl_gen_task_views 1
-- or
p_utl_gen_task_views @p_drop_only = 1

-- Create all task views and include encrypted PK
p_utl_gen_task_views  0, 1
-- or
p_utl_gen_task_views  @p_include_encrypted_pk = 1
```

1.6.4 p_amc_utl_gen_cdo_views

This stored procedure is used to generate views for the all CDO definitions.

It returns the same columns as the view v_cdo_Entities except that it filters by the specific CDO definition type.

Syntax

```
p_utl_gen_cdo_views
[   [ @p_drop_only = ] 'drop_only' ]
[ , [ @p_include_encrypted_pk = ] 'include_encrypted_pk' ]
```

Arguments

[@p_drop_only =] '*drop_only*'

This is a flag to indicate that the views should be dropped or created. '*drop_only*' is BIT. 1 = Drop the views and do not create them, 0 indicates that the views will be dropped and re-created. The DEFAULT value is 0.

[@p_include_encrypted_pk =] '*include_encrypted_pk*'

This is a flag to indicate that the generated views should include a column for the encrypted primary key (task_instance_id). '*include_encrypted_pk'* is BIT. The DEFAULT value is 0 (do not created encrypted primary key column).

Return Code Values

0 (success) or 1 (failure)

Remarks

The views are named with the following syntax:

v_cdo_<system_name>

where *system_name* is the system name of the cdo definition.

For example:

v_cdo_LoanApplication - All 'Loan Application' CDOs

The columns returned in each view included the following fixed and variable items:

cdo_instance_id	- Fixed
cdo_instance_id_encrypted	- Optional (based on flag used when generating the view)
ID	- ID of persisted entity (Can be NULL)
EntityName	- The name of the CDO property as configured.
DataObjectCategory	- The Category of object. Could be CDO
DataObjectType	- The Type of the object
ParentSystemName	- The system name of the parent in the hierarchy
CREATED_WHEN	- Fixed
CREATED_BY	- Fixed
UPDATED_WHEN	- Fixed
UPDATED_BY	- Fixed

Examples

-- Generate all cdo views

p_utl_gen_cdo_views

```
-- Drop all cdo views
p_utl_gen_cdo_views 1
-- or
p_utl_gen_cdo_views @p_drop_only = 1

-- Create all cdo views and include encrypted PK
p_utl_gen_cdo_views  0, 1
-- or
p_utl_gen_cdo_views  @p_include_encrypted_pk = 1
```

1.6.5 p_amc_utl_gen_search_views

This stored procedure is used to generate views on the indexed search and unique attributes for the following entity types: Parties, Repeating Attributes, Contracts, Product Definitions, Engagement Actions, Milestones, Case and Significant Events.

Syntax

```
p_utl_gen_search_views
[   [ @p_drop_only = ] 'drop_only' ]
[ , [ @p_include_encrypted_pk = ] 'include_encrypted_pk' ]
[ , [ @p_cbe_type = ] 'cbe_type' ]
```

Arguments

[@p_drop_only =] '*drop_only*'

This is a flag to indicate that the views should be dropped or created. '*drop_only*' is BIT. 1 = Drop the views and do not create them, 0 indicates that the views will be dropped and re-created. The DEFAULT value is 0.

[@p_include_encrypted_pk =] '*include_encrypted_pk*'

This is a flag to indicate that the generated views should include a column for the encrypted primary key (task_instance_id). '*include_encrypted_pk'* is BIT. The DEFAULT value is 0 (do not created encrypted primary key column).

[@p_cbe_type =] '*cbe_type*'

This is the type of cbe to generate search views for. The default value is NULL which means that views for all of the entity types search fields will be created. If specified it can be one of the following values:

Value	Description
1	Party
2	Repeating Attribute
3	Contract
20	Engagement Action
39	Significant Event
90	Milestone
91	Case
96	Product Definition

Return Code Values

0 (success) or 1 (failure)

Remarks

The views are named with the following syntax:

vs<entity identifier>_<entity_name>_<attribute_name>

where *entity identifier* is a value specific to each entity and follows the naming convention used in the DataMart table names. The values are:

p	Party
ra	Repeating Attribute
c	Contract
ea	Engagement Action
ev	Significant Event
mi	Milestone
ca	Case

pd Product Definition

entity_name is the system name of the entity

and *attribute_name* is the system name of the searchable attribute.

For example:

vsp_Individual_Surname - Search for Parties of type Individual by Surname

vsc_Loan_Reference - Search for Contract of type Loan using Reference number

The columns returned in each view included the following fixed and variable items:

<entity>_id - Variable – Always provided but the name varies by entity

<entity>_id_encrypted - Optional (based on flag used when generating the view)

<searchable column name> - Entity specific. The name of the searchable attribute.

RDG - If the searchable attribute is reference data, the RDG value

RDI - If the searchable attribute is reference data, the RDI value

Examples

```
-- Generate all searchable attribute views
p_utl_gen_search_views

-- Drop all searchable attribute views
p_utl_gen_search_views 1
-- or
p_utl_gen_search_views @p_drop_only = 1

-- Create all searchable attribute views and include encrypted PK
p_utl_gen_search_views  0, 1
-- or
p_utl_gen_search_views  @p_include_encrypted_pk = 1
```

To open the database view definition editor right click on your newly created database view and select "Edit...".

1.7 Guidelines On Running Reports

In this section we would like to provide some guidance on how to use views on Operational database.

In general we do not recommend running reports on operational database during business hours but we have many options to get close to real time data from operational database.

Bank can adopt that suites their business requirements.

Reference: https://technet.microsoft.com/en-us/library/ms187875(v=sql.110).aspx

Replication

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency.

SQL Server supports three types of replication, we recommend transactional replication.

Transactional Replication

Transactional replication typically starts with a snapshot of the publication database objects and data. As soon as the initial snapshot is taken, subsequent data changes and schema modifications made at the Publisher are usually delivered to the Subscriber as they occur (in near real time). The data changes are applied to the Subscriber in the same order and within the same transaction boundaries as they occurred at the Publisher; therefore, within a publication, transactional consistency is guaranteed.

If you choose to run reports on production database you will see CPU usage and Disk IO go up while running reports, depending upon the requested aggregated data.