Portrait
Foundation

# Batch Loading Framework Operational Guide

Edition 2.4

13 February 2013

Pitney Bowes
Software

# Portrait Foundation
# Batch Loading Framework Operational 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 is a division of Pitney Bowes Inc. (NYSE: PBI).

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

| **UK** | **America** | **Norway** |
|---|---|---|
| Portrait Software | Portrait Software | Portrait Software |
| The Smith Centre | 125 Summer Street | Portrait Million Handshakes AS |
| The Fairmile | 16th Floor | Maridalsveien. 87 |
| Henley-on-Thames | Boston, MA 02110 | 0461 Oslo |
| Oxfordshire, RG9 6AB, UK | USA | Norway |
| Email: support@portraitsoftware.com | Email: support@portraitsoftware.com | Email: support@portraitsoftware.com |
| Tel: +44 (0)1491 416778 | Tel: +1 617 457 5200 | Tel: +47 22 38 91 00 |
| Fax: +44 (0)1491 416601 | Fax: +1 617 457 5299 | Fax: +47 23 40 94 99 |

# About this document

## Purpose of document

This document is as a guide for the operation of the Batch Load Framework for bulk loading data into the Portrait metadata driven database. Details of how to configure of the Batch Load Framework are in the Batch Load Framework Configuration Guide.

## Intended audience

This document is intended for use by anyone who plans to bulk load data into the Portrait metadata driven operational database.

## Related documents

Batch Load Framework Configuration Guide

## Software release

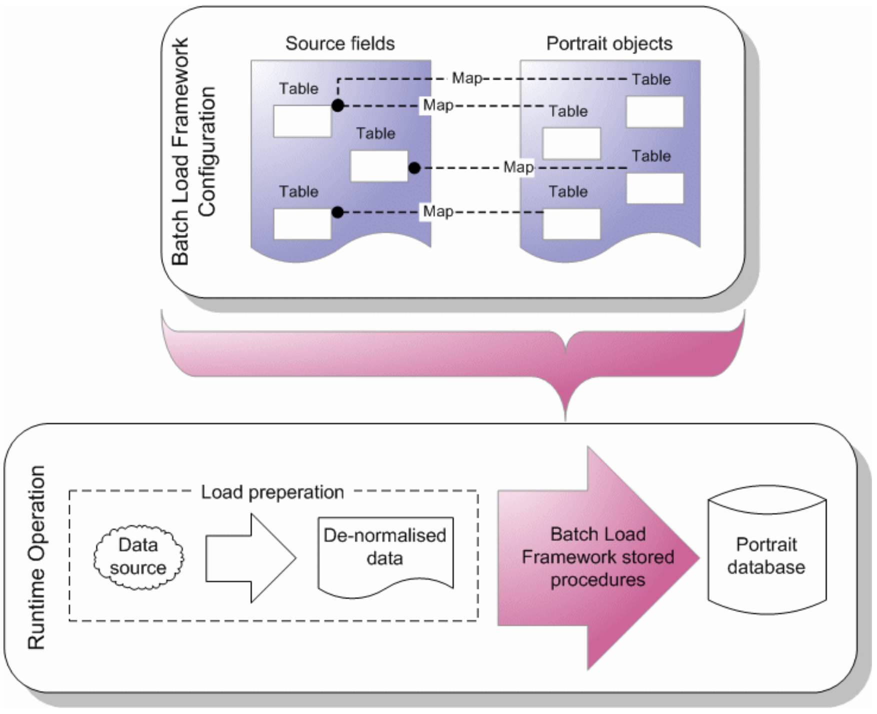
Portrait Foundation 3.2 or later.

# Contents

# 1 Overview of Batch Load Framework

The Batch Load Framework (BLF) provides a facility for bulk loading data into the Portrait Foundation metadata driven operational database. The BLF can be installed using the standard configuration suite and InstallShield scripts.

Figure 1 – Batch Load Framework architecture



The BLF process is in two distinct parts; configuration, done through the Configuration Suite, and batch-load of de-normalised data at runtime. This document is as a guide for batch-load of de-normalised data at runtime. It assumes that all configuration of the BLF is complete and de-normalised data has been prepared.

Details of how to configure of the BLF are in the Batch Load Framework Configuration Guide.

## 1.1 Main features

There are many features of the BLF the following are specific to runtime.

### Runtime

During runtime, the following features make the process easier.

- Chunking allows source data processing in smaller pieces to keep transaction sizes small and aid bad row diagnosing.
- Full support for stopping and restarting the load during batch execution.

### Error handling

When batch loading the data the following features help with processing of errors.

- Auto re-processing of failed chunks until the detection of the failed row.

- Failed rows moved to error table.
- Full auditing of batch and chunk instances through SQL queries of Portrait Foundation database tables – start and end times, status and error codes.

# 2 Batch load process

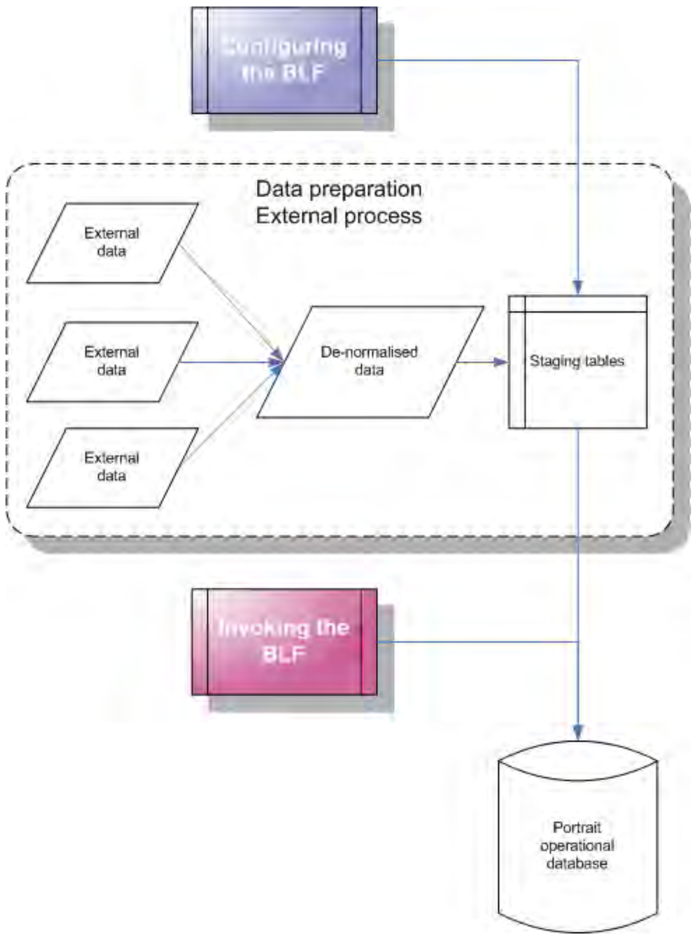
The batch loading process can be split into three parts.

1 Configuring the BLF to deploy staging tables and conversion procedures - The BLF configuration is done through the Configuration Suite and defines the relationships and transformations for the external data so that it can be placed into the Portrait Foundation database. These definitions are deployed to the database where a staging table and load procedures are stored. The staging table is a de-normalised view of all objects involved in the load and reflects the structure of the source file.

2 Preparing external data so that it can be loaded into the staging tables - Once staging tables have been created data from the data source can be loaded into them ready for the BFL procedures to load the data into the Portrait Foundation database at runtime.

3 Invoking the BLF procedures to load the staging table data into the Portrait Foundation database - The BLF procedures can be invoked, using the command line, Microsoft SQL Server Integrations Services (SSIS) or any other suitable tool, to pre-process and load the data from the staging table into Portrait Foundation database. Any errors that occur are written to an error table identifying the row that failed. In addition, auditing tables are populated for reporting on each batch instance.

Figure 2 – Batch load process



## 2.1 Operating a batch load

This document explains how to load the data into the Portrait Foundation metadata driven operational database at runtime, including creation, initialisation and error checking.

The guide assumes that the BLF has been configured (see the *Batch Load Framework – Configuration Guide* for further details) and de-normalised data has been loaded into the Portrait staging tables, and the table have been truncated. It is beyond the scope of this guide to explain how this is done, as there are many ways that you can do it.

The general mode of operation is for a job configured within SQL Server to execute the batch load stored procedure during periods of low database usage, for example overnight.

## 2.2 BLF process

The job can be split into six logical steps:

1. Truncate the staging table.
2. Load data into the Portrait Foundation staging table.
3. Perform any customer specific pre-processing.
4. Pre-process the staging table.
5. Execute the main load.
6. Check the batch error table and to determine the success of the load.

Data preparation must be done using third party tools, such as Microsoft SQL Server Integration Services (SSIS). The SSIS can model the runtime loading of the staging table and invoke the BLF runtime procedures.

# 3 Running a batch load

The runtime elements of the BLF are stored within the Portrait Foundation operational metadata driven database. The BLF is controlled with an API comprising of several stored procedures.

**NOTE**: The following examples use Microsoft SQL Server 2005 Management Studio Query Analyser tool for illustration purposes. Other methods such as command line or Microsoft SSIS can be used to load staging tables into the Portrait Foundation metadata driven operational database.

## 3.1 Creating a batch load

Execution of the BLF is through a batch load stored procedure that will load the data from the staging table into the Portrait Foundation operational database.

1 Open Microsoft SQL Server Management Studio 2005.

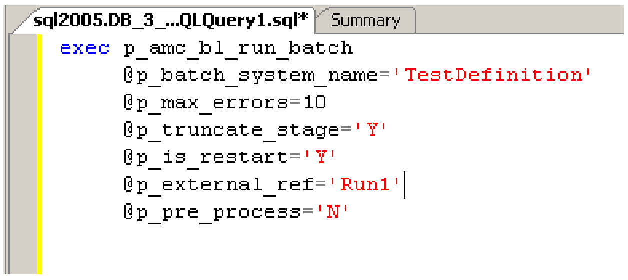
2 Select the staging table created and populated when configuring the BLF. To configure the BLF see the Batch Load Framework Configuration Guide.

3 Select the menu items File | New | Query with Current Connection.

4 Write a small batch file to load the staging table data into the Portrait Foundation operational database tables. See *Batch load commands and API* for API commands.

5 Save your script.

Figure 3 - Example batch load stored procedure



```
exec p_amc_bl_run_batch
     @p_batch_system_name='TestDefinition'
     @p_max_errors=10
     @p_truncate_stage='Y'
     @p_is_restart='Y'
     @p_external_ref='Run1'
     @p_pre_process='N'
```

## 3.2 Initiating a batch load

1 Open the batch loading script.

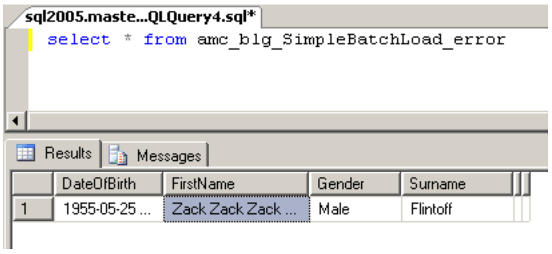
2 Select **Execute** from the **Query** menu.

3 Staging data in now loaded into the Portrait Foundation operational database.

## 3.3 Error checking the batch load

The batch load process automatically determines an erroneous row in the staging and loads this data into an error table for further analysis. You can analyse the error table using an SQL query.

Figure 4 – SQL query showing errors in the error staging table



sql2005.maste...QLQuery4.sql*

```
select * from amc_blg_SimpleBatchLoad_error
```

| | DateOfBirth | FirstName | Gender | Surname |
|---|---|---|---|---|
| 1 | 1955-05-25 ... | Zack Zack Zack ... | Male | Flintoff |

# 4 Batch load commands and API

The following information is a reference to the commands and API related to the operation of the BLF once Portrait Foundation staging tables have been created.

## 4.1 Stored procedures

The execution of the load is initiated and controlled with the following stored procedures:

| Stored Procedure | Description |
|---|---|
| `p_amc_bl_run_batch` | Executes or resumes a batch load including the pre processing |
| `p_amc_bl_stop_batch` | Halts the execution of a batch load |
| `p_amc_bl_preprocess_batch` | Executes the pre-processing stages of the batch load |
| `p_amc_bl_upd_batch_settings` | Update the settings for a batch definition |
| `p_amc_bl_process_ext_rdis` | Loads any missing external RDIs into the reference data item tables from the staging table. This needs to be run before any pre-processing is performed. |

## 4.2 Monitoring the batch load

The status of the main loading stage of the framework can be monitored with the `amc_bl_batch_instance` table:

| Status | Description |
|---|---|
| RUNNING | Batch is running. |
| STOPPED | The batch has stopped. |
| COMPLETED | The batch has completed. |
| ABORTED | The batch aborted because the maximum error count has exceeded. |

The `amc_bl_batch_instance` table can also be used to monitor the progress; the total number of errors, the number of rows processed, and the total number of rows are recorded for each load.

If the batch load fails before the main loading operation commences, that is the external reference uniqueness check fails, there is no entry in the batch instance table to record against the error. In this circumstance an error is raised by the `p_amc_bl_run_batch` stored procedure.

## 4.3 Configuring load execution

### 4.3.1 Settings for the batch definition

A number of settings can be configured on a batch definition basis using the `p_amc_bl_upd_batch_settings` stored procedure.

### Batch System Name

| | |
|---|---|
| @p_batch_system_name | VARCHAR |

Identifies the system name of the batch you wish to run.

### Use Unique External References

| | | |
|---|---|---|
| @p_use_unique_external_refs | VARCHAR | 'Y' / 'N' |

An optional external reference can be assigned to a batch instance, to ensure data is not loaded more than once into the operational tables. By enabling the uniqueness check, a batch load will fail if a duplicate external reference is found.

### Default Chunk Size

| | | |
|---|---|---|
| **@p_default_chunk_size** | INT | default is 1000 |

The framework loads the records into the Portrait Foundation database a *chunk* at a time. The chunk size is configurable and applies across the load. Increasing the chunk size will shorten the duration of the load, but may introduce table locking problems.

### Perform Updates

| | | |
|---|---|---|
| **@p_perform_updates** | VARCHAR | 'Y' / 'N' |

By enabling the update feature, records being loaded that already exist within the Portrait Foundation operational database are updated. Disabling this feature will result in these records being rejected as errors.

## 4.3.2 Settings for the batch instance

The parameters if the `p_amc_bl_run_batch` stored procedure can be used to control the execution of a batch.

### Batch System Name

| | |
|---|---|
| @p_batch_system_name | VARCHAR |

Identifies the system name of the batch you wish to run.

### Maximum number of Errors

| | |
|---|---|
| @p_max_errors | INT |

The maximum number of errors for the load can be specified so the framework will terminate when the value is reached. This feature is useful for failing an entire load after, thus saving unnecessary database processing. The default value is null, which sets no maximum, so the load will not be terminated.

### Truncate Staging Table

| | | |
|---|---|---|
| **@p_truncate_stage** | VARCHAR | 'Y' / 'N' |

The staging table can be optionally truncated at the end of a successful batch load. The truncation command is an efficient method of clearing the table, but must be disabled if the staging table is being populated with new data while a load is running.

### Is Restart

| | | |
|---|---|---|
| **@p_is_restart** | VARCHAR | 'Y' / 'N' |

If the batch load has been manually stopped, the load can be resumed by setting the parameter to 'Y', other wise the framework will restart the load from the beginning.

### External Reference

| | |
|---|---|
| @p_external_ref | VARCHAR |

An optional external reference can be assigned to a batch instance, to ensure data is not loaded more than once into the operational tables. If the uniqueness check is enabled, the load will fail when a duplicate external reference is found.

### Pre Process

| | | |
|---|---|---|
| **@p_pre_process** | VARCHAR | 'Y' / 'N' |

Pre-processing the staging table can be performed as separate step in advance, this may, or as part of the main load operation. To switch pre-processing on as part of the main load operation, set the parameter to 'Y'.

## 4.4 Errors

Runtime errors are reported by the BLF in two ways:

### Raised Errors

When the BLF is unable to process any records, an error is raised by the `p_amc_bl_run` stored procedure. Examples of this include; failure of the uniqueness check, the batch definition cannot be found, or the job is already running.

### Recorded Errors

When the BLF encounters a bad row the error is recorded and the framework continues to run. The bad row is moved to the error table. The errors code and primary key are also stored in the error table. The information about the error itself is stored in the table amc_bl_chunk_instance.  Examples of the cause of this error include; bad reference data, duplicate records, and command timeout.

If the maximum number of errors is specified the load will terminate when that number is reached.

Any scheduled task executing the BLF should record raised errors, detect recorded errors, and alert an operator as required. When recorded errors have been corrected, the records can be copied back into the staging table. The `p_amc_bl_clear_error_table` stored procedure clears the error table.

## 4.5 Application Program Interface

### p_amc_bl_clear_error_table

Clears error records from the error table

| | | |
|---|---|---|
| @p_batch_system_name | VARCHAR | System name of batch |
| @p_batch_instance_id | INT | Optional batch instance id |
| @p_external_ref | VARCHAR | Optional external reference |

- If the batch instance ID is supplied, all errors for the batch will be deleted.
- If batch instance ID is null, and the external reference is supplied, all errors for that reference will be deleted.
- If neither are provided, the error table will be truncated.

## p_amc_bl_run_batch

The backbone of the batch load runtime - controlling the load of the data and calling the other stored procedures as necessary

| | | |
|---|---|---|
| @p_batch_system_name | VARCHAR | System name of batch |
| @p_max_errors | INT | Optional maximum number of errors before load aborts. Null indicates no maximum. |
| @p_truncate_stage | VARCHAR | Optional parameter to truncate the staging table at the end of the load. (Y/N) Default is N. |
| @p_is_restart | VARCHAR | Optional parameter to indicate a batch restart. (Y/N) Default is N. |
| @p_external_ref | VARCHAR | Optional external reference |
| @p_pre_process | VARCHAT | Should pre-processing be performed at this stage. |

## p_amc_bl_stop_batch

Halts the loading process of the batch

| | | |
|---|---|---|
| @p_batch_system_name | VARCHAR | System name of batch |
| @p_batch_instance_id | INT | Optional batch instance id |
| @p_external_ref | VARCHAR | Optional external reference |

## p_amc_bl_upd_batch_settings

Updates the runtime properties of a batch

| | | |
|---|---|---|
| @p_batch_system_name | VARCHAR | System name of batch |
| @p_default_chunk_size | INT | the default chunk size for the batch |
| @p_use_unique_external_refs | VARCHAR | whether to use unique external references or not |
| @p_perform_updates | VARCHAR | whether to perform updates on existing records or not |

## p_amc_bl_run_preprocess

Executes the pre-processing stages of the batch load

| | | |
|---|---|---|
| @p_batch_system_name | VARCHAR | System name of batch |

### `p_amc_bl_process_ext_rdis`

Loads any external RDIs that are present in the staging table into the reference data table prior to loading the staging table.

If the RDG system name is not specified then any external RDGs that are used in the batch definition will be processed and missing values added. To limit the groups that will be processed call the procedure multiple times specifying the name of the RDG to process.