FastStats PeopleStage Technical Overview

A Technical Overview of the FastStats PeopleStage Architecture and Components

Prepared By: **Adam Robertson**

Last Modified:29/04/2015

Version: 2

FastStats™
Marketing Data Analysis Software

Apteco Limited, Tink-a-Tank House, 21 Jury Street, Warwick, CV34 4EH, UK
T: +44 (0) 1926 407565
E: support@apteco.com

W: www.apteco.com

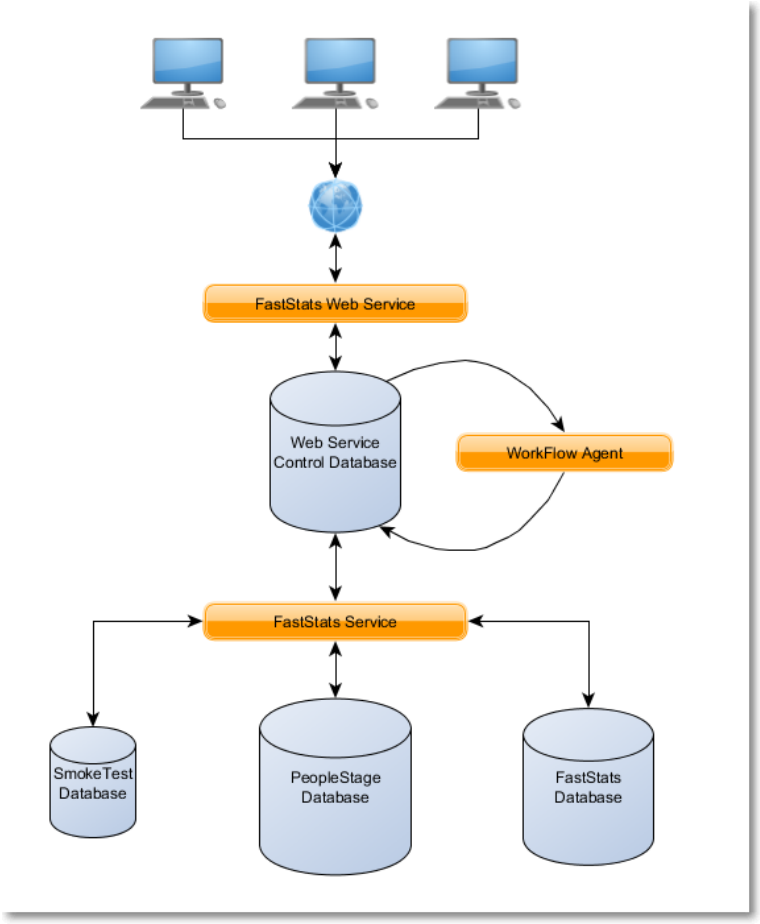
Contents

1 Introduction

This document outlines the components and architecture of FastStats PeopleStage. PeopleStage uses some of the same components as the rest of the FastStats suite of products, enabling it to share the same security, reliability and performance characteristics as FastStats Discoverer and FastStats Weblink, whilst containing some unique elements that provide the always-on marketing machinery that lies at the heart of PeopleStage.

1.1 Architecture diagram

The following diagram roughly lays out all the components of FastStats PeopleStage.



User machines running the FastStats PeopleStage client (typically delivered via the PeopleStage Launcher) communicate with the FastStats Web Service over a network such as The Internet or an intranet. The communication mechanism and the FastStats Web Service itself is the same as that used by Discoverer, WebLink and Excelsior. The PeopleStage Launcher is the same as its Discoverer counterpart – more details on how this work are available in the accompanying document titled “FastStats Discoverer Launcher Technical Details”.

The FastStats Web Service places jobs submitted by the client into a job queue within the Web Service Control Database which are then processed by the FastStats Service. The FastStats Service then uses the FastStats Engine and FastStats Database built by FastStats Designer (the FastStats system) to process requests and place the results back into the Control Database ready for retrieval by the PeopleStage client via the Web Service. Again, this is the same mechanism as used by other FastStats products.

However, the FastStats Service also contains a PeopleStage module which can read and write to the

main PeopleStage Database as well as a “Smoke Test” test database. It can also place jobs into a section of the Web Service Control Database which cause the PeopleStage WorkFlow Agent to run various marketing operations such as identifying an audience, sending out email broadcast messages or attributing responses on user-defined schedules.

The PeopleStage database can be said to have 3 main sections:

A “model” section that holds information about the PeopleStage diagram shown to the user.
A “schema” section that holds corresponding information about all live PeopleStage campaigns in a format that the FastStats Service can use.
A “history” section, which records the history of where people have flowed through the diagram and a communications history of what was sent to each person for every campaign.

Further details on the PeopleStage database can be found in Appendix B.

1.2 The Lifecycle of a typical PeopleStage campaign.

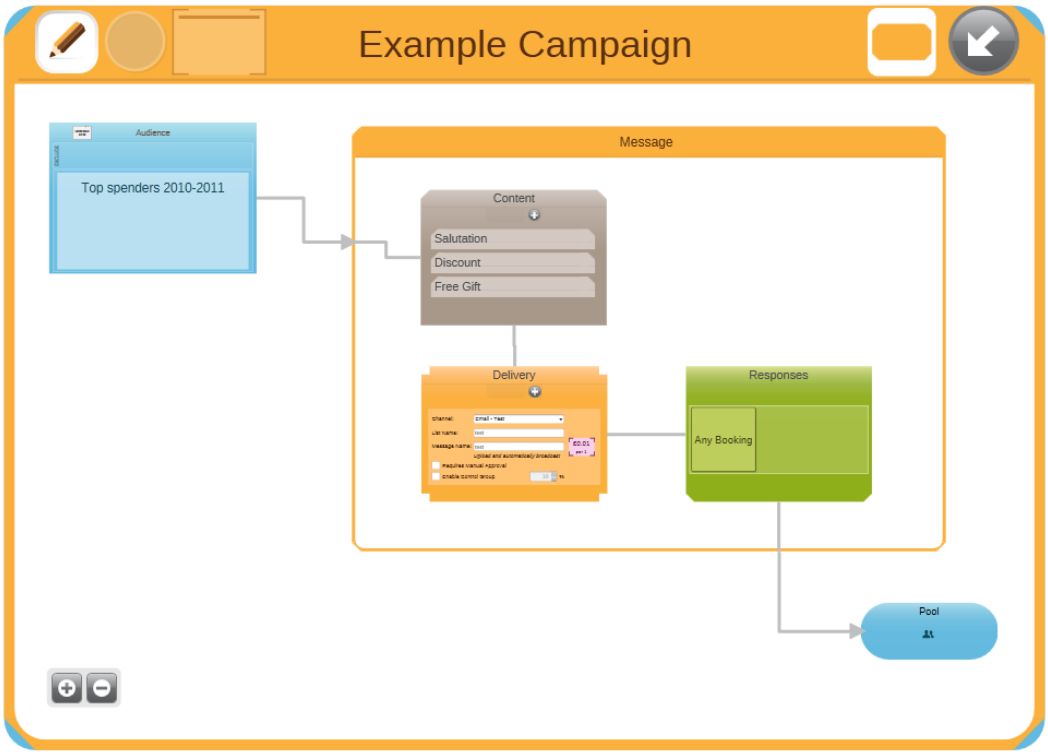
Within PeopleStage, users define marketing campaigns and then “publish” them, causing them to be processed. A PeopleStage campaign might typically involve identifying a target audience, specifying some content to send to them and then outputting a text file for a mailing house or uploading data to one of the many third-party email broadcasters supported by FastStats. PeopleStage will then monitor email activity and be able to report on email responses as well as being able to report on behavioural responses (such as new orders or inquiries) caused by the marketing activity.

Each part of defining and running a PeopleStage campaign involves various components within the architecture and the following sections will discuss which components are used at each step.

For more information on the detail of creating and publishing a PeopleStage campaign, see the training documentation or online help.

1.2.1 Creating a campaign

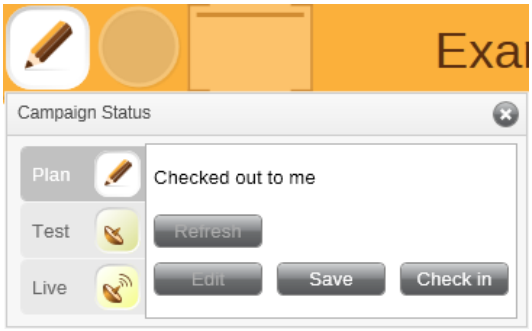
The first step in the process is to create a PeopleStage campaign. A typical campaign would consist of an audience definition and a message definition followed by a results pool where those that have been sent the message will be held. A typical message itself would consist of a content definition describing how the message will be tailored for each member of the audience, a delivery definition describing which channel to use to deliver the message and a response definition, describing what kind of behaviour constitutes a response to the campaign.



The campaign is defined within the PeopleStage client, using tools and predefined templates from the PeopleStage library as well as selections created with FastStats Discoverer and variables from the FastStats system.

1.2.2 Saving and loading the campaign

Whilst a campaign is being drafted but before it is ready to be published it can be saved and reloaded. Because PeopleStage is a multi-user system, campaigns must also be “checked out” and “checked in” by the user to obtain and relinquish an exclusive lock on the editing of the campaign to stop other users making conflicting changes. Saving, reloading (termed “Viewing”), checking in and checking out (termed “Editing”) are all done from the control at the top left of the campaign.



Pressing these buttons send commands from the PeopleStage client via the Web Service and Control Database to the FastStats Service, which updates and retrieves the campaign (as part of the wider PeopleStage diagram) in the model section of the PeopleStage database.

1.2.3 Testing the campaign

Once the first draft of a campaign is finished, it can be published to the smoke test database. This will allow the user to see how the people selected in the audience step will flow through the campaign and how the various content variations will be applied. The smoke test database is a copy of the schema

and history sections of the main (live) PeopleStage database which can then be discarded when a test is finished. Changes to the smoke test database won't affect the schema or history sections of the live PeopleStage database and publishing a campaign to test won't send a real email broadcast, instead using a built-in dummy broadcaster.

Other than this, the mechanism for publishing and running a test campaign is the same as that for a live campaign, which is described in the publishing to live section, below.

1.2.4 Modifying the campaign after testing

Based on the results of a test run, the campaign may be altered and tested further by iterating through the previous steps until a final draft of the campaign is created. This can then be published to live.

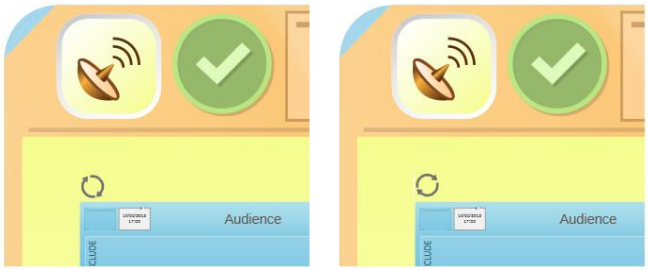
1.2.5 Publishing the campaign to live

Once a campaign is ready to be run properly, it can be published to live. When a campaign is published to live, the PeopleStage client sends a command through the Web Service to the FastStats Service to do the following:

Save the campaign to the model section of the PeopleStage database.
Write further information about the structure of the campaign to the schema section of the PeopleStage database
Write information about how to run the campaign to the "workflow" section of the Web Service Control database, ready for the Workflow Agent to process. This would include information such as when to run the audience step, and then which steps to run after this has finished (which in the above example would be the content step) and so on.

The Workflow Agent is a separate Windows service, and when it detects that the schedule for a step in the campaign becomes due it looks up the job specified in the workflow section of the Web Service Control database and submits it back to the job queue in the Control database ready for the FastStats Service to pick up. The FastStats Service will then pick up the job and process it given information from the FastStats system and the schema section of the PeopleStage database. The results of the job will usually either be an upload of records to a third-party email broadcaster or for information about the flow of people to be recorded in the history section of the PeopleStage database.

The PeopleStage client continually polls the Web Service for information on any currently running PeopleStage jobs. The Workflow Agent monitors the progress of the jobs that it has sent to the FastStats Service and stores this information in the workflow section of the Control database. The Web Service reads this information and returns it back to the client. This allows the client to indicate which jobs are currently being processed with a rotating progress indicator.



Optionally, a FastStats Email Response Gatherer can be configured to retrieve response information from the third-party email broadcaster. This will record responses in a separate Email Response database which can then be queried by the FastStats Service so that audience steps can be defined later on in the campaign to select all people that opened or clicked on a particular message.

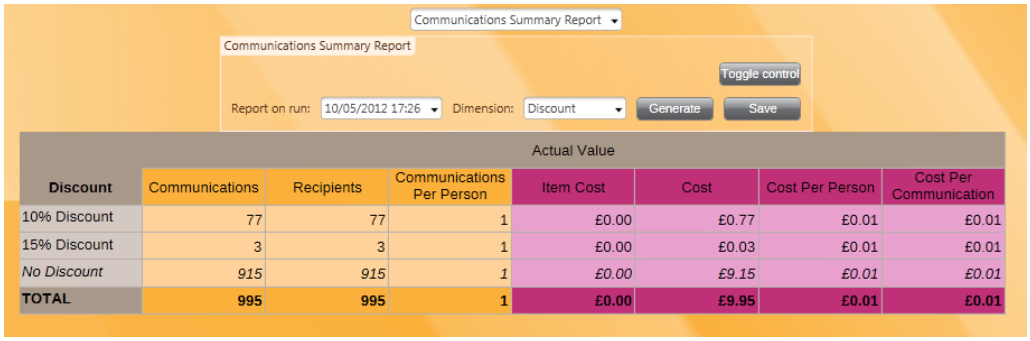
1.2.6 Gathering behavioural responses over time

Behavioural responses to the campaign (such as people making further inquires or purchases based upon an email) are handled in PeopleStage by recording them in the FastStats system. This is done by capturing the response in FastStats Designer from the user's online transaction database or data warehouse and building it in to a transaction table in the FastStats system. If there is a key code or communication key associated with the transaction then this too should be built into the FastStats system. PeopleStage is then configured with the FastStats variable names for these pieces of information.

When the FastStats system is rebuilt, the Workflow Agent will detect this and automatically run a response attribution process. This analyses any new transactions in the FastStats system and records any that match the rules defined in the responses step of the campaign into the history section of the PeopleStage database. Behavioural responses can then be used in the same way as email responses to select people in further audience steps that met certain behaviour rules. Information on these responses can also be viewed in PeopleStage reports.

1.2.7 Reporting on campaign performance

PeopleStage offers various reports for campaigns which are based on information held in the history section of the PeopleStage database, the Email Response database and the FastStats system.



	Actual Value						
Discount	Communications	Recipients	Communications Per Person	Item Cost	Cost	Cost Per Person	Cost Per Communication
10% Discount	77	77	1	£0.00	£0.77	£0.01	£0.01
15% Discount	3	3	1	£0.00	£0.03	£0.01	£0.01
No Discount	*915*	*915*	*1*	*£0.00*	*£9.15*	*£0.01*	*£0.01*
TOTAL	**995**	**995**	**1**	**£0.00**	**£9.95**	**£0.01**	**£0.01**

When a report is requested, the PeopleStage client sends a request to the FastStats service which can potentially query the history section of the PeopleStage database, the separate Email Response database or the FastStats system (depending on the report) to build up the report and returns the results.

PeopleStage can also be configured to create reports in Excel on a recurring schedule and send the results to a distribution list. In this case the report definition is created in the client, sent to the FastStats Service and stored in the Web Service Control database. Then when the FastStats Service scheduling mechanism (the pre-existing mechanism used to run scheduled tasks defined in Discoverer) detects that a report is due to be distributed, the FastStats Service again queries the databases mentioned above and then builds up and distributes the Excel report.

1.2.8 Extending the campaign

Once the campaign has run and the reports viewed, the user may then want to either schedule the campaign to run periodically or to extend it by adding extra steps to send more messages to the people that responded or behaved in particular ways.

2 Behind the Scenes

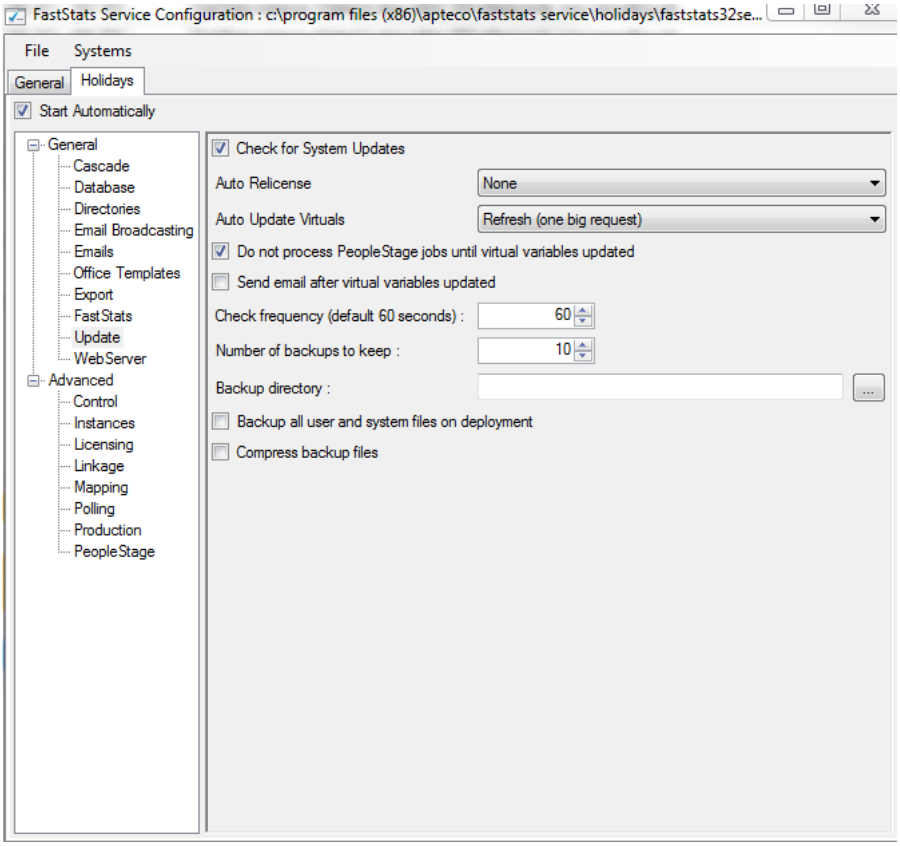
2.1 Normal Operation

2.1.1 Process Flow

The diagram in Appendix 3 describes the normal flow of events from a user publishing a Campaign to the Workflow picking up individual actions as they are due to be run, and submitting them to the job queue.

2.1.2 Held Job Types

PeopleStage jobs can be put on hold after a deployment until all virtual variables are refreshed through use of the HeldJobTypes table in the WebService database. This is necessary when a VV is needed in a PeopleStage campaign either in an Audience selection or to be output by a Delivery step.

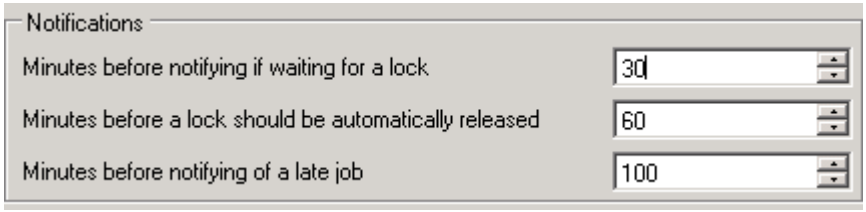


2.2 FastStatsService Configuration (to Handle Errors)

2.2.1 Setting Up Notification Options

There are some central notification options which govern all PeopleStage jobs. In the future, users will be able to set their own through the PS interface.

The values below are sensible defaults:



Locks are applied to the PeopleStage database to enforce the constraints that a user can set within the diagram, such as “1 Campaign started per Week”, to avoid multiple actions that are running concurrently on different threads both consuming the quota set in the constraint. The first action to run will apply a lock to the database; subsequent actions whose running could be affected have to wait for the lock to be released.

The above settings would notify the user who had published the diagram that their campaign had been waiting for 30 minutes due to another action having locked the database. After a total of 60 minutes the lock would be released and the second job would be allowed to proceed in parallel.

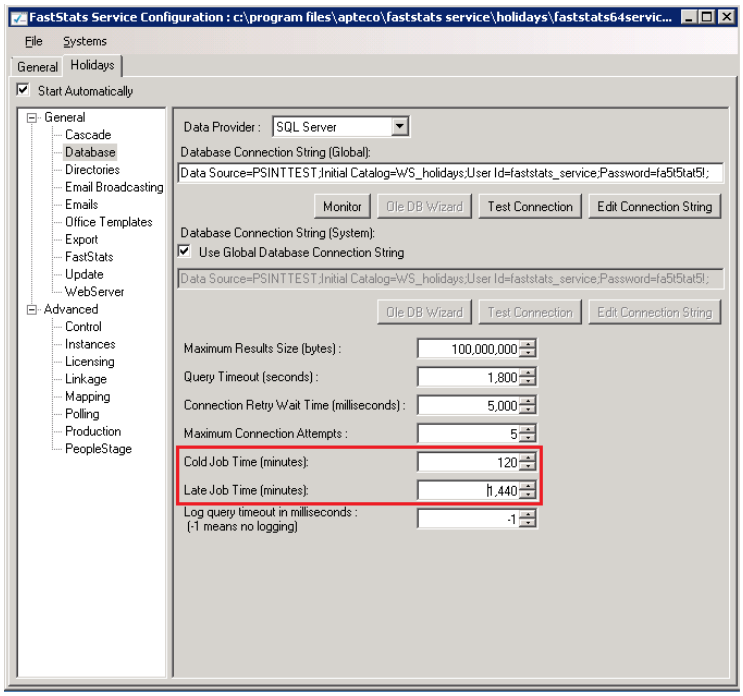
The Workflow is constantly monitoring for diagram steps that need to be run, and put the associated jobs in the job queue as they become due. The FastStatsService will process the job as soon as it is able to, but if over 100 minutes (in the above example) has elapsed then the user who published the Campaign will be notified that the job is late. The job will, however, still be run. The following section deals with how to cancel a late job.

2.2.1.1 Setting Cold and Late Job Times

The settings below control the time at which a job will be declared too Cold or Late to be processed. A Late job is one that has been waiting too long in the job queue without being processed. A Cold job is one that is being processed but has taken too long to complete.

The settings below would:

- Terminate a job that had been running for 120 minutes, declaring this job as Cold.
- Refuse to process a job that had been in the queue for 1440 minutes (24 hours) declaring it as Late.



Note that the previous section only controlled notifications, and so a job that was 100 minutes late would still be run, given the settings above.

2.2.2 Setting Up the Workflow Options

The Workflow retry options control what happens next when a job has been declared Cold or Late etc. Again, future functionality within the interface will make this control simpler.

The settings below will retry a job which is declared Cold or Late up to a maximum of 2 times. After this the job will be declared as having errorred.

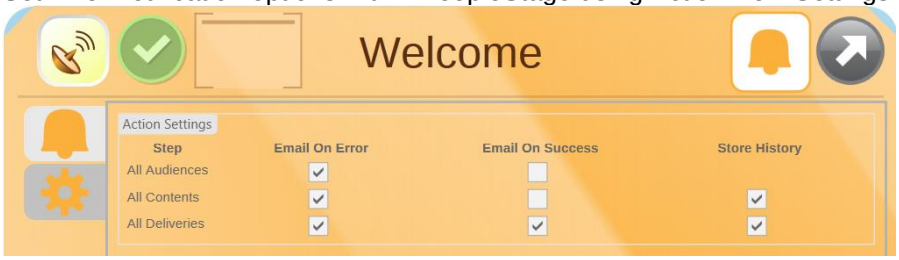
The “PauseGroupRetryOnResume” option will pause a campaign (aka group) whenever an error occurs within the campaign (or when the retry attempts are exhausted). The action which errorred will be “Retried On Resuming” the campaign.

Jobs are typically cancelled if they are running when the FastStatsService is restarted. The settings below will also retry these jobs a maximum of twice.

2.2.3 Setting Notification Options in PeopleStage

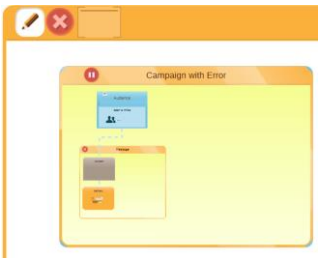
Use FastStatsService Configurator to set notification options for locks and late jobs (not so essential).

Set Error notification options within PeopleStage using Action view Settings.



These are on a per user basis and email the FastStats user who is logged in to PeopleStage. They are currently committed by Publishing the Campaign.

2.3 Diagnosing an Error

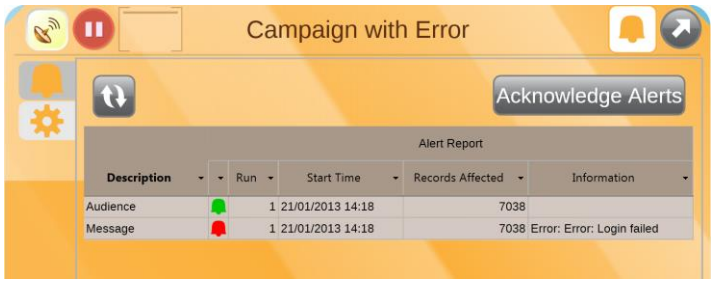


2.3.1 Alert Indicators and Action History

When an error occurs within a Campaign:

- The Campaign with an error shows a red Pause.
- The site of the error is shown with red X.
- Outer Areas show escalated red X.

Details of the error message can be found by clicking on the error indicator:



2.3.2 Using the Workflow Monitor

The Workflow Agent Monitor **ActionState** table lists each Action (i.e. step in the diagram), gives details of those which are currently running and summarises the last run of each action. Further **JobSummary** information can be found for each individual action.

The **Job Status** column lists each job which is "**Ready**" to run (i.e. has reached its scheduled time). The Workflow Agent will submit a number (as determined by the "Number of Threads" workflow option) of actions to the job queue. At this point a **FastStats Job Id** is displayed.

The Job Status changes to "**Running**" when the FastStatsService has picked up a job. All PeopleStage jobs are issued with a priority of 100 so typically only a few threads are able to run PS jobs. Details are shown in the Operation column and a red graphical indicator is shown in the Progress column.

The JobQueueStatus column will show **Done** when the FastStatsService has finished processing the job.

ActionState

Description	FastStats Job ID	Job Status	Last Started	Last Finished	Type	Source	Retries	JobQueueSt
Block details for Audience		Ready	21/01/2013 14:18:08		PS_BLOCK	PeopleStage => Marketing Processes => Insurance Division => Test Area => Campaign with Error...	1	
Pool			21/01/2013 14:16:50	21/01/2013 14:16:50	PS_FLOW	PeopleStage => Marketing Processes => Holidays Division => Engagement Area => Special Offer...	0	
Pool			21/01/2013 14:07:24	21/01/2013 14:07:25	PS_FLOW	PeopleStage => Marketing Processes => Holidays Division => Engagement Area => Special Offer...	0	
Delivery	93462		21/01/2013 14:06:12	21/01/2013 14:12:59	PS_FILEBROA...	PeopleStage => Marketing Processes => Holidays Division => Engagement Area => Special Offer...	0	done

??? If an Error occurs or a job goes cold – doesn't say Errorred, but can see that the job didn't finish.

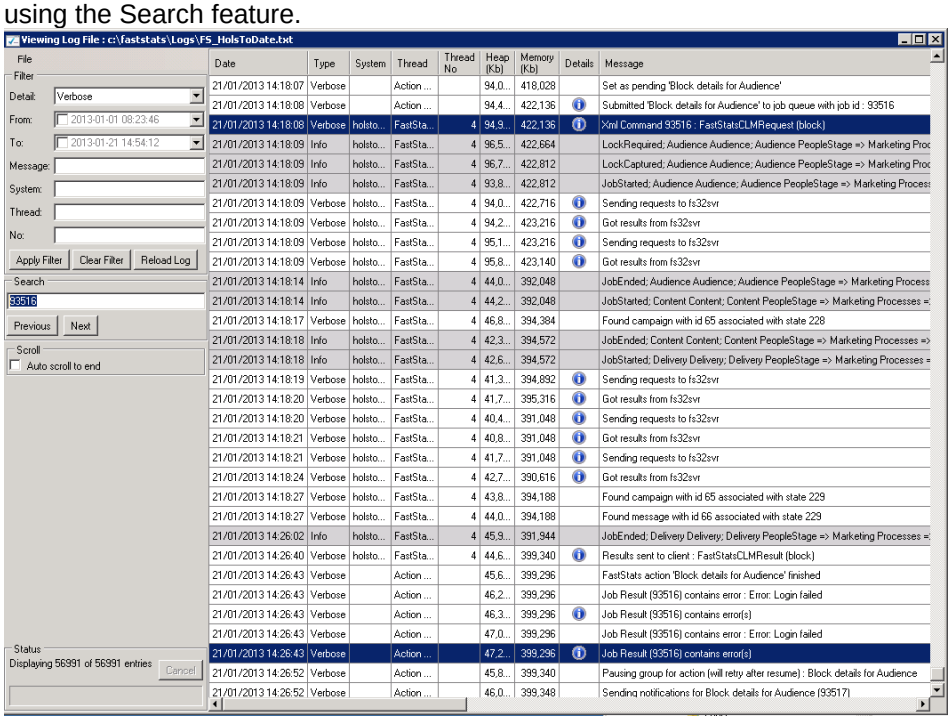
Further details can be found by double clicking on an entry in the **ActionState** table to see the **JobSummary** information. Here you will see a count of Errors and Warnings, as well as a history of previous runs.

JobId	ActionType	ActionDescription	ActionSource	StartTime	FinishTime	Errors	Warnings
93516	PS_BLOCK	Block details for Audience	PeopleStage => Marketing Processes => Insurance Division => Test Area => Campaign with Error => Audience	21/01/2013 14:18	21/01/2013 14:26	1	0

You may need to refer to the FastStatsService log to find further information. Note the **FastStats JobId** from either the ActionState or JobSummary tables.

2.3.3 Using the FastStatsService Log

Find the entries in the log relating to the JobID that you identified using the Workflow Monitor – e.g. by using the Search feature.



Filter by thread to see all the log entries for the particular thread that ran the Job.

Filter by date to see what other threads were doing at the time.

Double click on the row saying XmlCommand to see the PeopleStage request.

2.3.4 Using the Web Service Database

You can get more information from the WebService JobQueue, for example, if the FastStatsService log has been truncated.

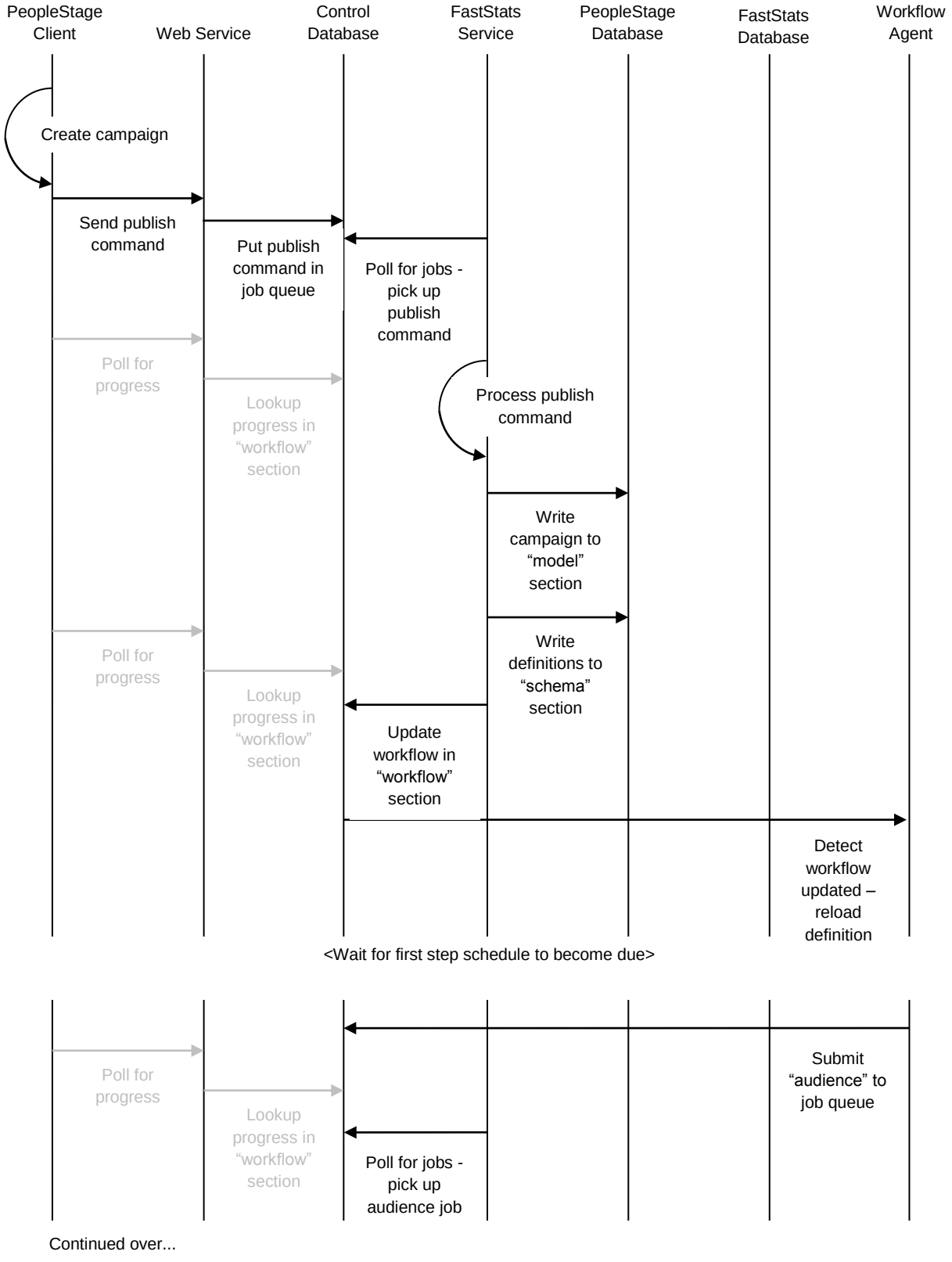
```
SELECT *
  FROM [WS_HolsToDate].[dbo].[JobQueue]
  where JobID = 93516
```

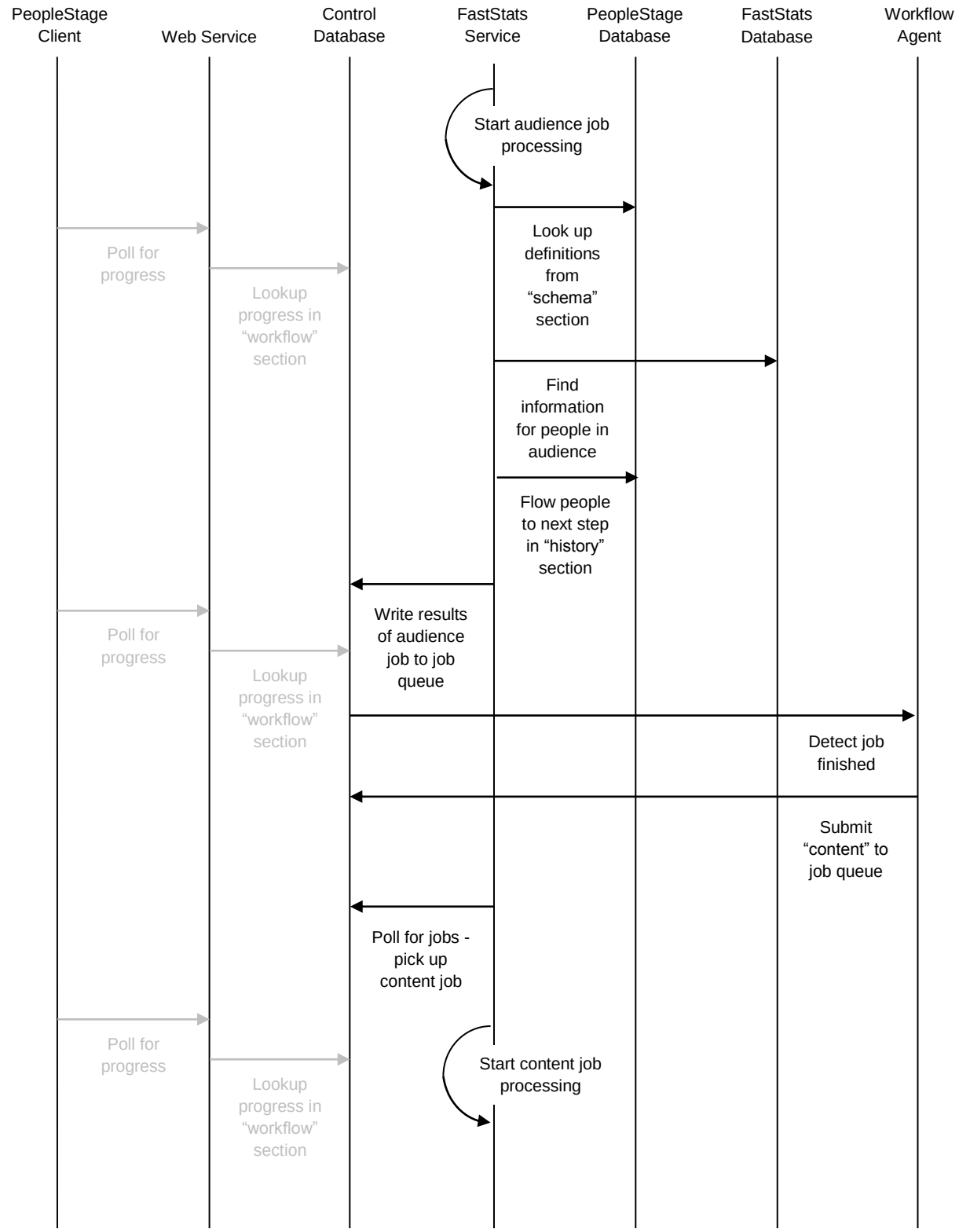
2.3.5 Using the PeopleStage Debug Log

The Debug Log shows errors that have arisen as an immediate result of a user action such as Publishing, running a report etc or as a result of UI processing such as getting progress information. It won't show errors that have arisen during the scheduled run of a Campaign.

3 Appendix A – Sequence diagram for publish to live process

The following diagram shows the sequence of communications and steps involved during the publishing and running of a campaign.




PeopleStage Client
Web Service
Control Database
FastStats Service
PeopleStage Database
FastStats Database
Workflow Agent
Start audience job processing
Poll for progress
Lookup progress in "workflow" section
Look up definitions from "schema" section
Find information for people in audience
Flow people to next step in "history" section
Write results of audience job to job queue
Poll for progress
Lookup progress in "workflow" section
Detect job finished
Submit "content" to job queue
Poll for jobs - pick up content job
Poll for progress
Start content job processing
Lookup progress in "workflow" section

4 Appendix B – PeopleStage Database Structure

The PeopleStage database can be said to have 3 main sections:

1. A “model” section that holds information about the PeopleStage diagram shown to the user.
2. A “schema” section that holds corresponding information about all live PeopleStage campaigns in a format that the FastStats Service can use.
3. A “history” section, which records the history of where people have flowed through the diagram and a communications history of what was sent to each person for every campaign.

Generally, the model and schema sections of the database will be very small compared to the history section of the database as they only contain two different views of “metadata” about the diagram, whereas the history section contains information for each person that has passed through any part of the diagram.

4.1 The Model Section

The model section is made up of all the tables that start with the name “Model”. These tables are read from to load the PeopleStage diagram and written to when parts of the diagram are saved or checked in or out.

The model section consists of a collection of elements, where each step in the diagram is a row in the **ModelElement** table. The relationship between child and parent steps (i.e. which steps are contained within a particular Message or Campaign) is recorded in the **ModelChildToParentElement** table.

Every time part (or all) of the diagram is saved, a new version is created (in the **ModelVersion** table) and every element in the diagram is associated with this version (using the **ModelVersionToElement** table). Note that even if only one Campaign is saved, when the new version is created every element in the whole diagram is associated with this new version, but only the elements that have changed are updated. This is achieved because the ModelElement table has an Id and Revision which make up the primary key, along with some XML that records the element’s definition. When an element is changed a new row is added to the ModelElement table using the same Id and an incremented Revision. This allows the model section of the database to hold details of all versions of the diagram that have been created. Administrators can view this information using the PeopleStage Model Database Inspector tool.

When parts of the diagram are checked in or out, the details (including time stamps and the user that performed the actions) are recorded in the **ModelElementCheckout** table.

Finally, when part of a diagram is published (to live or test) as well as being saved, the diagram is labelled with the text “Publish” (for publishing to live) or a string based upon the campaign’s id (for publishing a particular campaign to test). This then allows the system to know what the “live” or “test” version of each element in the diagram is. This is done with the **ModelLabel** and **ModelLabelToElement** tables.

See the document “PeopleStage – Overview of the PS Database and FastStats system” for further information on the PeopleStage database.

4.2 The Schema Section

When a diagram is published, as well as saving and labelling data in the model section of the database, data is updated in the schema section of the database. The schema section holds the information that the FastStats Service uses to know how to move people through the parts of the diagram and how to send the appropriate messages to email broadcasters, etc. Note that the FastStats Service doesn’t use data in the model section of the database at all when actually running particular actions or steps.

Data for the schema section is calculated from information held in the model section by the FastStats

Service during a publish action. Examples of tables in the schema section include **StateDefinition**, **ActionDefinition**, **TreatmentDefinition** and **TreatmentLibraryItem**. See the document "PeopleStage – Overview of the PS Database and FastStats system" for further information on the PeopleStage database and which tables constitute the schema section.

The schema section is also referenced in the template FastStats design used to incorporate PeopleStage data into a FastStats system.

When a publish takes place, a new copy of the schema data is calculated in memory within the FastStats Service and compared to the information held in the database. Any changes that are then required to bring the database up to the same state as the newly calculated schema are then made. Note that any new information is inserted into the appropriate schema table, but out of date information is not deleted using a SQL DELETE statement, but marked as deleted in the **DeletionControl** table instead. SQL Views are then used to only select rows containing data that hasn't been deleted. For information that has only been modified, the old row is marked as deleted in the DeletionControl table and then a new row is inserted.

Also, note that some pieces of information (specifically descriptions and structural information) are updated in the **DescriptionHistory** and **StructureHistory** tables using a CreatedOn date. The most up to date piece of information for a particular id will be the one with the latest CreatedOn date.

There are no relational connections (such as foreign keys) between the schema section and model section of the database. However, some ids from the schema section of the database are held within the XML definitions saved in the ModelElement table.

4.3 The History Section

The history section of the database contains details for how each person that has been processed by PeopleStage has progressed through the diagram. The table where the most detailed information is held, and the table that will be the largest in the whole PeopleStage database, is the **StateHistory** table. This contains a row per person per state that they have entered, where a state (as defined in the StateDefinition table in the schema section) exists before and after every action (as defined in the ActionDefinition table) performed by PeopleStage. Most PeopleStage actions have a corresponding step in the PeopleStage diagram, but only some states have a corresponding pool in the diagram. The rest are not visible (marked as SystemDefinied in the StateDefinition table) but still exist as holding areas before and after each action.

The other tables in the history section contain summarised information based upon the state history table, such as the **ActionHistory** or **Communication** tables. The ActionHistory table holds information about the number of people processed by each action and when the action ran. The Communication table holds information about when data was output or uploaded to an email broadcaster.

Along with the schema section of the database, the history section is also referenced in the template FastStats design used to incorporate PeopleStage data into a FastStats system.

Finally, note that unlike the model section, tables in the history section of the database do have SQL foreign key relationships with tables in the schema section.

See the document "PeopleStage – Overview of the PS Database and FastStats system" for further information on the PeopleStage database.

4.4 Smoke Test Databases

PeopleStage allows campaigns to be published to "Test" as well as "Live". This is done by having a separate test database to keep any test run information isolated from the main live database. The test database contains tables from the schema section and history section, as it is information in the schema section that allows the FastStats Service to process PeopleStage actions and the results are

stored in the history section. The model section does not need to be isolated from the live database and so isn't present in a smoke test database.

When a campaign is published to test, new tables are created in the test database with names of a format "<tablename>#<smoke test instance id>" (i.e. ActionDefinition#22 or StateHistory#35). Then depending on the table, data is either copied from the live database into the new smoke test version or it is combined with the live version of the table using a SQL UNION operation. Then any new information added into these tables doesn't affect the operation of the live database and when the test is finished the tables can be deleted.

4.5 PeopleStage Lifecycle - Database Perspective

The PeopleStage campaign lifecycle described in section 1 of this document can then be shown with details of how each section of the database is affected:

1. Creating a campaign

Model	-
Schema	-
History	-

No changes to the database – all work done in memory within the PeopleStage client

2. Saving and loading the campaign

Model	Changes loaded & saved to model section
Schema	-
History	-

3. Testing the campaign

Model	Data loaded from model to perform publish
Schema	Test database created, publish writes schema information
History	Results of running actions saved to history in test database

4. Modifying the campaign after testing

Model	Modifications saved back into model section
Schema	-
History	-

5. Publishing the campaign to live

Model	Data loaded from model to perform publish
Schema	Publish writes schema information in main database
History	Results of running actions saved to history in main database

6. Gathering behavioural responses over time

Model	-
Schema	-
History	-

No changes to the PeopleStage database – responses stored in FastStats system

7. Reporting on campaign performance

Model	-

Schema	-
History	Results queried from history to populate reports.

8. Extending the campaign

Model	Modifications saved back into model section
Schema	-
History	-