

# Toolkit & Demonstrator Final Report Template

<b>Project name/acronym:</b>	LIPID
<b>Project website/blog address:</b>	<a href="http://www.lipidproject.org.uk">http://www.lipidproject.org.uk</a>
<b>Report, author(s):</b>	Colin Dalziel
<b>Contact person</b> (if different from above):	
<b>Date:</b>	20 September 2006

## Methodology

The project largely followed the original project plan (Appendix 1) with the only significant difference being the level of involvement of Brockenhurst College who were the original developers of the Sweet toolkit. This was due to the original developer of the toolkit having left the college and there was no one available with a suitable level of expertise to support the project. The initial project included some involvement from Maven Associates who were intending to support some of the issues relating to the use of the SITS student management system. Maven were available to research the functionality of the toolkit and offer the support we had planned for Brockenhurst therefore the funding originally allocated to Brockenhurst was actually reallocated to Maven Associates.

## Implementation

The approach taken involved a fairly lengthy research phase where the project team investigated the capabilities of the Sweet toolkit, possible routes for extracting data from SITS and what was required within PebblePad to meet the needs of a number of case scenarios. Details of the research and initial planning phase are available from the LIPID project website (<http://www.lipidproject.org.uk>) in the project requirements specifications documentation. (Attached as Appendix 2) This report also includes the technical design, testing and maintenance strategies. As the project comes to a close we are still trying to implement the work at Queen Margaret's University College. The slight difference in the academic year has meant it has been difficult to gain access to key staff at Queen Margaret's however the college remains committed to the project and will implement the toolkit over the coming weeks. The progress of this implementation will be available from the project website at <http://www.lipidproject.org.uk>.

## Outputs and Results

The key output from the project is a modified version of the Sweet.Net toolkit with details of the Format for the data exported from the student management system. All of these items together with detailed project information is available from the project website <http://www.lipidproject.org.uk>.

The project work is now being implemented on an institutional basis with every student at The University of Wolverhampton being able to draw upon their institutionally held data from within their personal ePortfolio space.

## Implications

The work undertaken forms a good basis for any group activity based upon module registration. Whilst the student management system (SMS) used in the project was SITS this is not a prerequisite. As the data exported from the SMS was in CSV format any system capable of supporting a basic query could be used to generate data in an appropriate format. Similarly as the data exported from Sweet.net is an Enterprise Web service it could be used with a number of systems including a VLE.

One of the current limitations of the process developed is the mass nature of the data processing. Essentially a complete set of data is imported on a periodic basis. To take the functionality to a higher level it is desirable for some extra work to be carried out to develop an event driven based update for the data held in Sweet.Net. It is envisaged this work will be investigated over the next academic year.

## Appendix 1 Project Plan

### LIPID: Linking the Institutional and Personal Information Domains Demonstration of SWEET.net

This project will implement the existing SWEET.net toolkit using the SITS student management system and the PebblePad eportfolio system as the communicating platforms.

The Project will run from February 1<sup>st</sup> 2006 to July 31<sup>st</sup> 2006.

All outputs from the project will be Open Source with free redistribution from the project website and appropriate JISC websites such as the ELF.

The project will include implementation at two host institutions:

- to help prove the robustness of the original SDK;
- to prove its portability and re-usability by others;
- to provide an authentic test bed for further development during the lifespan of the project.

The common services **Group**, **Member**, **Person** will be proven and extended to include **Role** in the management of eportfolio gateways.

### Context

ePortfolios have rapidly occupied the foreground of innovative approaches to learning, assessment and personal development: though all conceptions of eportfolios are far from innovative.

Eportfolios differ from other elearning systems, such as VLEs, in that they are more concerned with the personal experience of learning than with an institutional concern for content delivery, subscription to services and tracking use. Within the eportfolio, and wider, development community discussions of 'learner data' tend to conflate institutional data and eportfolio data as an amorphous whole. These data types should be considered separately and treated separately, whilst recognising that they need to work in harmony to support claims of learning; of progression; of professional competence and of aptitude for new roles.

Linking the institutional data domain with the personal data domain remains challenging; an issue made more difficult by protective data gatekeepers. Until data management issues mature middleware services like SWEET.net provide an opportunity for learners to access institutionally held (and approved) data to their personal learning systems for onward use in online CVs; applicational presentations; and career portfolios. Notwithstanding obvious, and longer term, concerns for the meaningful authentication of qualifications, certificates, licences and transcripts; an immediate case for data sharing is the avoidance of unnecessary repeated data entry by the user e.g. name; address; contact details and courses studied. Opportunities for further data transfer and mechanisms for authenticating institutionally approved data may feature in this project through the development of the **Role** service.

SWEET.net has been developed to include group or affiliation data; this data is valuable in securing the space between the individual and the institution. In the case of PebblePad this information would be used to manage teacher controlled 'gateways' which allow eportfolio users to 'publish' work for institutional purposes. The use of gateways provides an obvious delineation between the personal eportfolio assets and those created for formal purposes (assessment, accreditation, monitoring). The use of group information held by the MIS, accessed by the SWEET.net service, and provided to the eportfolio system, allows teachers to manage multiple/diverse/large groups of learners without a burdensome administrative overhead.

In short, the implementation of SWEET.net in the space between the institutional MIS and the personal eportfolio presents an opportunity to gain efficiencies for both the learner and the teacher easing their association with both and reducing potential barriers and objections to wider adoption of eportfolio systems.

## Integrating SITS and PebblePad

### **About SITS...**

SITS:Vision is a management information system primarily targeted towards Higher Education institutions in England, Wales and Scotland and the Further Education sector in Scotland. SITS:Vision is currently used by over 60% of the UK HE market and its customers include; University of York, University of Bath, University College London and University of Warwick.

### **About PebblePad...**

PebblePad is an ePortfolio system developed within Higher Education but now in use in all education sectors and in professional CPD. Though only recently developed and released PebblePad is in use in two JISC-funded projects: other customers include City University London, Coventry University, University of Gloucestershire and Queen Margaret University College.

This toolkit implementation project will utilise the Group, Member, Person services developed in SWEET.net to fetch information from SITS to populate the area of PebblePad called 'About Me'.

About Me contains typical learner data including name(s), contact details, courses, affiliations and richer claims about experience and learning preferences. The services listed will be used to populate appropriate fields and, where possible, provide a service to update the MIS record from authorised changes within the eportfolio.

The service **Role** will be developed from the existing services to provide group information to control access and permissions on eportfolio gateways i.e. teacher/institution controlled spaces where eportfolio assets can be published for formal/structured processes.

## Rationale

Management Information Systems are ubiquitous and SITS is the sector leader. However, many are 'homegrown' and the demonstration of an open source toolkit to support system integration, both proprietary and otherwise is highly desirable.

ePortfolios will increasingly dominate the HE learning landscape in the near future. Both systems provide information about learners; some is institutional and some is deeply personal. A web service that is able to link both domains is not only desirable but increasingly essential. Funding for this implementation project will not only prove the portability and usability of the toolkit but also identify areas for further development including work around eportfolio repositories; federated asset management and application processes/services.

It is hoped that the involvement, in an essentially technical project, by representatives of the teaching community, registry and IT services; working in collaboration with a dynamic spinner company across the boundaries of University; University College and FE College – and indeed across national borders – makes this a truly integrative proposal.

Whilst such a broad spectrum of stakeholders might appear over-ambitious for a project of this scale it is the very willingness and commitment to success of the partnership which makes this a viable proposition.

## Methodology and Work Plan

We will adopt a test-driven approach with broad stakeholder involvement to deliver a robust implementation project.

We will implement the toolkit in the primary host institution (University of Wolverhampton) to test the toolkit's extant reusability.

We will develop the toolkit to improve its portability and further evaluate it in a second host institution (QMUC).

Throughout the project we will develop the toolkit as a reusable service commensurate with the purposes and standards of the JISC and the Open Source community.

### **The following standards will be employed**

IMS ePortfolio 1.0  
IMS Enterprise Spec  
Web services

### **The following technology will be employed**

.net 1.1 and MS-SQL for server side development

The Web Service Layer, Exemplar Implementation and server-side admin tools will be further developed in C# using the Visual Studio.net 2003 IDE. The services will be hosted by the institutions on Windows Server 2003 but will also be compatible with Windows 2000 and Windows XP Professional.

### **Intellectual Property Rights and Sustainability**

Project outputs will be made available, free at the point of use, to the UK HE, FE and school communities in perpetuity, and they will be disseminated widely by the project in partnership with JISC, and through other e-learning network organisations.

Software components of the deliverables will be released under open source licences without restriction to ensure that they be freely shared with the open source and wider development community. All software that is developed will be made available free of charge to that community in perpetuity and all code developed will be made available through open source models. Outputs will be licensed under an Open Source agreement that promotes maximum reuse.

The project will undertake, subject to available funding, to continue the extension and enhancement of the toolkit according to IMS LIP and Enterprise specifications and other appropriate standards, and continue to make such developments available to the JISC community.

## Work Plan

1. Collaborate with SDK developer to establish current development status and develop working methodology
2. Scope specific detailed functionality for all services
3. Develop integration plan and refine toolkit as necessary
4. Develop quality testing plan for the service based upon use cases with appropriate partner involvement
5. Install, evaluate, develop toolkit – iterative process (University of Wolverhampton)
6. Interim evaluation of the project to-date to prioritise enhancement features and opportunities for further development work
7. Implement and evaluate toolkit (QMUC)
8. Terminal evaluation
9. Adhere throughout to the OSMM guidelines in particular to contribute to the development of both the toolkit and the developer community **(Deliverable)**
10. Collaborate with the toolkit developer to update all code, installation scripts, documentation and other deployment artefacts **(Deliverable)**
11. Publish service to ELF **(Deliverable)**
12. Write report with recommendations for future work **(Deliverable)**
13. Demonstrate ‘live’ use of the toolkit **(Deliverable)**

## Workflow

	Week	2	4	6	8	10	12	14	16	18	20	22	24	26
Item 1 - 2														
Item 3 - 4														
Item 5														
Item 6														
Item 7														
Item 8														
Item 9 - 10														
Item 11 - 13														

## Stakeholder Analysis

<b>Stakeholder</b>	<b>Interest / stake</b>	<b>Importance</b>
Learner	Requires institutionally held information for reuse in support of eportfolio outputs. Ease of access to stored data provides efficiencies of use sustaining engagement	High
Teacher	Requires group information to effectively manage the space between the institutional demands on eportfolio and the personal nature of the system. Ease of administration improves persistence in use	High
JISC	Evaluation of a funded toolkit and exploration of innovative, shifting developments	High
CETIS	Feedback on the IMS EWS spec	Medium
Institutions	Innovation, collaboration, networking with other project teams. Enhanced systems interoperability and experience to support future planning	Medium
Vendors, developers, integrators	Scope for interoperability of existing applications. Explicit support for open standards	Medium
Registry	Movement toward distributing data and challenging the hegemony of data silos	Low

## Appendix 2

### LIPID Sweet.Net Interface Requirements and Design

<u>Software Requirements</u> .....	8
<u>Introduction</u> .....	8
<u>Purpose</u> .....	8
<u>References</u> .....	8
<u>Overall Description</u> .....	8
<u>Scope</u> .....	8
<u>Batch Import interface and Activity Report</u> .....	9
<u>Event Driven Import Interface</u> .....	9
<u>System Interfaces</u> .....	10
<u>User Interfaces</u> .....	10
<u>Batch Interface Report Content</u> .....	10
<u>Event Interface Audit Log</u> .....	11
<u>Configuration Data</u> .....	11
<u>Operational Environment/Constraints</u> .....	11
<u>Availability and Recovery Handling</u> .....	11
<u>Recommendation</u> .....	12
<u>Performance Requirements</u> .....	12
<u>Service Level Objectives</u> .....	13
<u>Security</u> .....	13
<u>Portability</u> .....	13
<u>Use Cases</u> .....	13
<u>Batch Refresh of Student Data held in Sweet.Net</u> .....	13
<u>Event Driven Student Data</u> .....	14
<u>Specification</u> .....	15
<u>Data</u> .....	15
<u>Group Information</u> .....	15
<u>Student Personal Information</u> .....	16
<u>Artefacts to be developed</u> .....	16
<u>Testing Strategy</u> .....	17
<u>Maintenance Strategy</u> .....	17
<u>Installation Activities</u> .....	17
<u>Pebble Learning/University of Wolverhampton Responsibilities</u> .....	17

# Software Requirements

## Introduction

### Purpose

This document identifies software requirements for a new interface that will be developed for the sweet.net technology. The interface will take a data feed generated by another system, process it and enable it to be loaded into sweet.net.

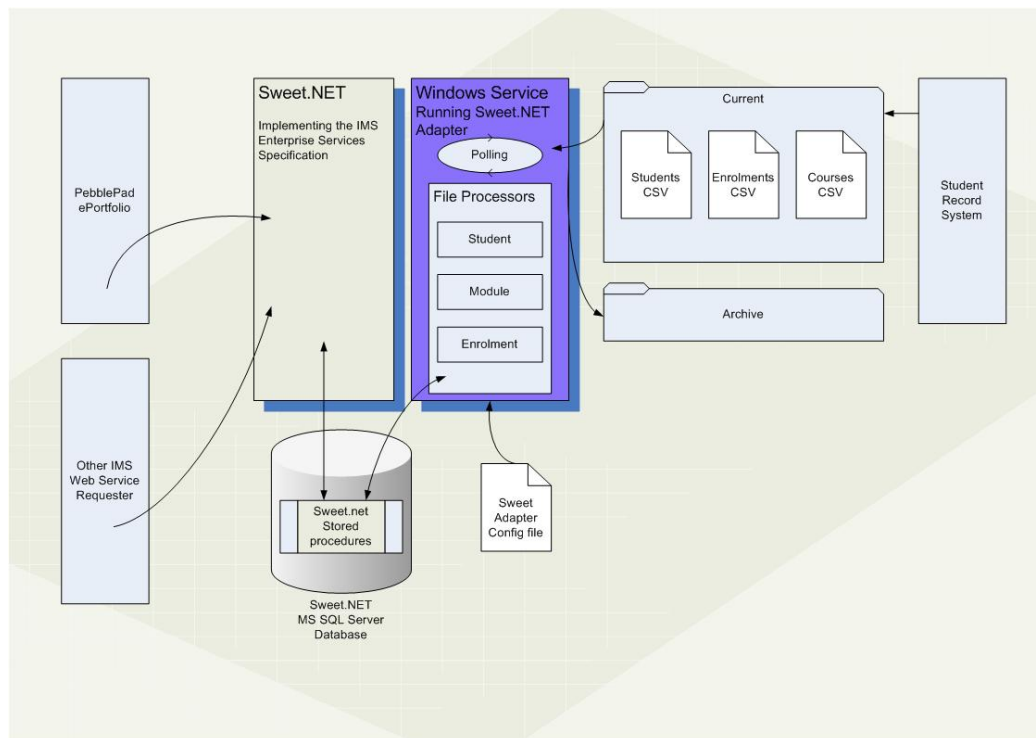
This document will be used by Pebble Learning and Maven Associates to agree what work must be completed for creation of this interface.

### References

Sweet.net (Enterprise Web Services with Timetable Extensions for .NET) - <http://www.brock.ac.uk/sweet>.

## Overall Description

While the solution implementation is independent of either the upstream student record system (SRS) or the downstream ePortfolio application the success of the process is dependant on creation of the data files from the SRS and their successful provision to the sweet.net interface.



### Scope

Two software artefacts will be produced:

- 1) Batch Import interface and Activity Report
- 2) Event Driven Import interface

### **Batch Import interface and Activity Report**

This interface will be used in an “Extract, Transform and Load” (ETL) style. The extract will be performed by a separate component, transformation requirements are expected to be minimal, and load functions will be performed using sweet.net functionality.

Because each invocation of this interface represents a complete refresh of the data supplied by the partner application the processing must handle all variations of create, update and delete processing. Deletes will only be apparent by their absence from the new dataset, so a reconciliation process is required.

The data will be formatted in a single file using a CSV structure. Data volumes are projected to be in the order of 20k records.

Frequency of invocation will probably be weekdays overnight, however higher frequencies should be possible. The frequency of the invocation will be determined by 2 factors: the frequency of the data drops and the interval set for the polling. The polling interval will be controlled by a configuration setting in the new interface. The frequency of the data drop will be reliant on the partner application.

The data provided by the SRS is deemed to be definitive therefore it will replace any matching data held in sweet. The sweet.net application is currently web service enabled and so can receive update/delete/insert requests from any client invoking these services. Therefore, the sweet.net application can be subject to several data feeds at any given time including during the period of the batch import. During the batch import process, the data in the CSV file will be considered definitive and will take precedence over any data provided by a web service.

In order to track processing a report should be generated whenever this import interface runs. The report will detail all actions performed and flag up any discrepancies. The report will be for human use as opposed to a feed into some other system and will not need to generate historical reporting data.

The batch import will also be used when the event driven interface is in use, though less frequently. In this scenario the discrepancies report would be likely to capture situations where an event failed to a) be triggered in the source application, b) transmitted to the sweet.net application, or c) handled completely by the event interface.

### **Event Driven Import Interface**

The event driven interface will complement the batch interface and allow the solution to operate in a timelier manner. The upstream source application will trigger when certain events occur and marshal together associated data items into an XML document. Some mechanism will be used to make these documents accessible to event driven interface.

Events will include create, update and delete. Data volumes will be lower than for the batch update as we only handle change events. Subject to the mechanism used to make these documents available to new interface they may be processed as they are created, or collected together and processed at some later time. A suitable naming convention will be used to simplify sequence handling.

As for the batch interface, the event interface is deemed compliant in that it accepts valid data and potentially replaces data already held by sweet.net.

## **System Interfaces**

Sweet.net currently does not offer a suitable framework to perform reliable scheduling or polling capabilities so it is envisaged that the solution will be an external application or windows service running outside the current sweet.net application.

ASP.Net does provide notification hooks to signal the start and end of an application through implementing the Application\_Start and Application\_End methods in Global.asax. However, these methods are not appropriate for starting, scheduling, or stopping a file poller for several reasons:

Asp.Net applications “end” when there are no active sessions

- Asp.Net applications “start” in response to a page request following the last “end”

Therefore it cannot be guaranteed that an ASP.Net application will be a running state at any given time unless it is servicing page requests. It also cannot be guaranteed as to the delay in a page request being issued to “start” the application following a shutdown.

The solution will be written in C#.Net using standard system classes for polling, threading and file io. The solution will also make use of the existing sweet.net application for database access and class definitions.

## **User Interfaces**

If the solution is delivered as a windows service, it will be possible to start and stop the service through a standard windows service property sheet. It is anticipated that changes to the configuration of the interfaces will be effected through configuration files so no additional UI is anticipated. The two sweet interfaces will be responsible for producing audit trails in the case of the event handler and a status report for the batch interface. In both cases the output will be written to standard text files.

### **Batch Interface Report Content**

The following list considers possible content for the batch import report.

- Standard details such as run date and time, hostname and identifying details, version information and such like.
- A summary detailing main outcomes of this run:
  - Inbound records processed
  - Number of creates, updates and deletes
  - If possible number of records held by sweet.net – this may be the same as the number of inbound records assuming only one feed into sweet, or greater if multiple feeds are implemented.
- Detail records showing a list of changes in a format such as:
  - Action ID Before After
- A list of unchanged records

## **Event Interface Audit Log**

The event interface should log its activities in some sort of audit log.

The format could be the same as the detail records of the batch report.

To be determined: the actual mechanism to use for logging audit data. A possible solution to consider is log4net – see <http://logging.apache.org/log4net/> for further details.

## **Configuration Data**

Both implementations will require some configuration data. This will be stored in an XML document.

Possible configuration data will include:

- File system locations for inbound data files
- File system locations for outbound reports and audit records
- File housekeeping task – see Availability and Recovery Handling
- Security attributes, though these will probably be as for sweet.net itself

## **Operational Environment/Constraints**

The following items need to be established.

- Host Computer Hardware
- OS version and support levels
- .Net environment versions
- Any known constraints with any of the above – e.g. disk space for storing data files, RAM and/or CPU etc.

## **Availability and Recovery Handling**

The interfaces themselves do not own any real production data. They consume data generated up stream from the SRS in the form of files, and after processing the responsibility for managing the inbound data is dealt with by sweet.net. Only supplemental data is generated by the interfaces for reporting and audit purposes. In general processing will be designed to process a file (batch or event), commit the changes in sweet, and then perform some sort of housekeeping function on the processed inbound file.

The primary concern here is with data integrity. It is assumed that operational procedures are in place for management of the SQL Server database such that backups are performed and logs are maintained. As a result any failures resulting in a system outage of some sort allow the system to be recovered to a known state with all data intact up to the point of failure.

For the batch interface it should be a relatively simple task to create operational procedures to ensure data integrity. For example, prior to performing an import complete a backup of the affected tables and import file(s) so that a failure during import can be resolved by restoration to a known state and re-run. This assumes that sweet.net is not in use at the time of import. More subtle approaches can be developed if needed.

For the event driven interface a more granular approach might be required. This is based on the assumption that the upstream SRS application will not be easily able to recreate the event detail XML documents. To ensure that an event file is not lost as a

result of a system outage of some sort (e.g. disk failure) and that the files can be re-presented to the event interface they must be retained somewhere on a redundant location. This could be on the SRS server, on a staging server, or on a separate physical disk on the sweet.net server. These files will need some sort of housekeeping.

There are a number of approaches to consider that could simplify the processing and operational procedures here:

1. Rather than attempt to manage recovery for the event driven interface at a very granular level, state that the operational procedures for recovery from a serious outage that may raise doubts regarding data integrity (could some event data have been lost in transit) then after recovery processing the batch interface should be run to re-sync the data. This would require an SRS operator to generate an extract file from the SRS for use in the batch interface.
2. Replace the transport method. Rather than use files consider use of a messaging approach, such as MSMQ or similar. Persistent messages that contain the event data will have the same level of integrity as the database. This would eliminate the housekeeping effort but would introduce some additional technology and management activities.

### **Recommendation**

There are many limitations in using files for integration, especially when using an event driven real-time approach. Rather than try to handle these in the short term instead use option 1 above which provides a satisfactory solution until service levels require a higher level of availability.

When availability requirements become more demanding it will be very beneficial to move to option 2.

### **Performance Requirements**

Early discussions indicate a need to “stress” test the solution.

We should be clear about the objective of any testing. Standard testing will include Unit testing (“do the new artefacts operate as designed?”) and Integration testing (“do they operate as designed when deployed with all other solution components?”). Such tests confirm the functionality and integrity of the solution.

It may be that formal stress testing is inappropriate. Stress testing is concerned with abnormal operation of the system – resources are demanded in uncharacteristic quantities and frequencies. Such testing is intended to push the solution operation beyond standard operational boundaries and would require additional effort to conduct.

Instead a level of performance testing is probably more appropriate. The primary objective would be to provide some indicative statements regarding expected throughput in a given environment.

The simplest approach to gathering this kind of performance data is to instrument the code such that it can generate processing statistics. The same approach as suggested for the Event Interface Audit Log would be a viable and configurable approach here. Using log4net would allow performance data to be collected based on configuration settings rather than code changes. This could add a little to the development effort.

## Service Level Objectives

At the current time any formal objectives are unknown, though the previous few sections have speculated about possible targets.

## Security

No special security processing is required. The solution will be delivered as a windows service and will be subject to whatever security model is used for managing services on the host platform. The solution will require io access to read and write to pre-specified directories.

## Portability

The developed code need only work in the current version of sweet.net. Any other considerations regarding code portability are as established for sweet.net.

## Use Cases

### Batch Refresh of Student Data held in Sweet.Net

#### Brief Description

The SRS generates an extract file that details group and personal information for all students. The file is made available to the sweet.net batch import interface and records are iteratively processed and stored by sweet.net in its database.

#### Actors

SRS – Student Record System  
Sweet.net

#### Triggers

The interface monitors a file system location and on arrival of a file matching an expected name format initiates import processing.  
In the event that sweet.net is inactive when the file arrives interface initialisation processing will find an unprocessed file and import it.

#### Preconditions

SRS creates and makes available the extract file.

#### Basic Flow

1. On arrival of an extract file the interface processes each record iteratively.
2. Records are first validated to ensure data matches expected formats.
3. With a valid set of records a reconciliation exercise is performed to identify a sequence of “create”, “update” and “delete” sweet.net operations.
4. Changes are applied to the sweet.net application and reporting data is collected. Additionally, optional logging events may be generated to capture performance data.
5. A summary report is written to a file (see Batch Import interface and Activity Report).
6. File housekeeping tasks are performed to remove the extract file from the import directory and place them in a temporary archive location. On the first invocation of the week interface will delete import files older then one week.

**Success Outcome**

A subset of SRS data is consistently stored in the sweet.net application.  
A processing activity report is available for review.

**Minimal Guarantees**

If the import file cannot be correctly processed then the process will terminate at step 3 above and include appropriate details in the report. Thus sweet.net will retain the current, though out of date, student data.  
If errors occur during step 4 these will be logged and the interface will continue to process remaining changes if possible.

**Exception Handling**

- Technical exceptions
- Business rule exceptions

**Notes, Open Issues****Event Driven Student Data****Brief Description**

Whenever the SRS processes a create update or delete for a subset of student personal information of group (module) data the SRS marshals required data together and generates an XML document stored in a file. This file is then (using a mechanism yet to be confirmed but assumed here to be file based) made accessible to sweet.net where the new interface handle import processing and makes changes to sweet.net data.

**Actors**

SRS – Student Record System  
Sweet.net

**Triggers**

The interface monitors a file system location and on arrival of a file matching an expected name format initiates import processing.  
In the event that sweet.net is inactive when the file arrives interface initialisation processing will find an unprocessed file and import it.

**Preconditions**

SRS creates and makes available the XML event file(s).

**Basic Flow**

1. On arrival of an event file the interface opens and validates the XML data.
2. Based on the data content a “create”, “update” or “delete” sweet.net operation is performed.
3. Audit/logging events are generated to capture processing activity to aid in possible problem resolution.

4. File housekeeping tasks are performed to remove the event file from the import directory and place them in a temporary archive location. The interface will delete import files older than one week based on some criteria to be determined.

#### Success Outcome

sweet.net is updated in with the event data.

#### Minimal Guarantees

Any error in the basic flow will be captured and logged such that an operator can easily find errors and will have access to information required for problem resolution.

#### Exception Handling

- Technical exceptions
- Business rule exceptions

#### Notes, Open Issues

1. File housekeeping.

## Specification

This section contains a “work in progress” that develops a specification for aspects of the developed artefacts. It will be developed by the joint contributions of Pebble Learning, Mike Beeston and the .Net developer.

## Data

The following data items are known to be required in Sweet.Net and so will be found in the inbound data.

Note that dependant on the use case in question the data will:

- Be in a different format:
  - For batch refresh CSV
  - For event XML
- Significantly, the event interface will be explicitly notified of deletes. This is not the case for batch refresh.

Sweet.net is concerned with storing data for use in satisfying requests for data in an IMS format. ACTION: A mapping and transformation analysis is required for the supplied data into sweet.net (assumed at this time to closely match IMS structures). Note that where a data name is noted in the source system for reference – ideally any dependencies (coupling) to source system should be kept to a minimum. The objective is to process a file of data, either in csv (use case 1) or XML (use case 2).

## Group Information

Group data describes information regarding course modules creation, scheduling and subsequent maintenance.

For full details on IMS group data definitions see

[http://www.imsglobal.org/enterprise/entv1p1/imsent\\_infov1p1.html#1427147](http://www.imsglobal.org/enterprise/entv1p1/imsent_infov1p1.html#1427147)

Other data may be added to this list.

<b>Description</b>	<b>Source Name</b>
Course Title	Course short name
Course ID	Course code
Module Title	Module Name
Module ID	Module Code
Occurrence	MAV Occurrence
Occurrence Time/Day	
Students Name	First Forename + Surname
Student ID	Student Code
Tutors Name	PRS_NAME.PRS
Tutor ID	

### **Student Personal Information**

<b>Description</b>	<b>Source Name</b>
ID	Student code
Title	Student title
First Name	Student first forename
Other names	Student second forename + Student third forename
Last Name	Student surname
Date of Birth	Student date of birth
University email address	Student University email address
Gender	Student gender
Address Home	Student home address line 1 + Student home address line 2 + Student home address line 3 + Student home address line 4 + Student home address postcode
Home Phone	Student home telephone number
Address contact	Student contact address line 1 + Student contact address line 2 + Student contact address line 3 + Student contact address line 4 + Student contact address postcode
Contact Phone	Student contact telephone number
Phone Other	Student home telephone number (second line)
Phone mobile	Student home telephone number (mobile)
Course Title	Course code
Previous Module Results	
Modules Currently studying	
Prior Academic qualifications	

### **Artefacts to be developed**

A set of .Net classes that will comprise a windows service to poll for files, perform transformations on the inbound data, and cause any changes to be applied to the datastore and logged.

Further analysis is required at this stage to identify exactly which classes will need to be developed.

## **Testing Strategy**

Pebble learning will create test data that accurately represents the inbound data structures for the use cases to be developed. This will be documented with details of any valid data that might be included and describe how it will be used by sweet.net. This data will be used off site to perform all unit and system testing on newly developed code.

Integration testing will be performed on site.

## **Maintenance Strategy**

As OSS software there will be no formal maintenance support. The code will be documented and available for re-use under some OSS licence. Currently sweet.net does not appear to be released under any such agreement. It may be appropriate for Pebble Learning to consider selection of a suitable licence – see <http://www.fsf.org/> and <http://apache.org/licenses/> for further details.

## **Installation Activities**

The sweet.net extensions will be released as OSS that can be easily integrated with the base sweet.net implementation. Installation guidance will be written that includes any pre-requirements and co-dependencies.

## **Pebble Learning/University of Wolverhampton Responsibilities**

1. Confirm approach presented in this document as acceptable.
2. Provide test data.
3. Facilitate access to sweet.net technical support should this be needed.
4. Provide an environment for integration and performance testing.
5. Validate that the delivered solution satisfies the requirements for import of SITS data into sweet.net.